MASTER EN SCIENCES INFORMATIQUES

Service Continuity Mode 2 in 5G Networks (Network layer version)

Dekeyser, Pascal

Award date:
2019

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR
Faculté d'informatique
Année académique 2018–2019

Service Continuity Mode 2 in 5G Networks (Network layer version)

Pascal Dekeyser

Maître de stage : Laurent Schumacher

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Laurent Schumacher

Co-promoteur :

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

À ma femme Alessia, mes enfants Matteo, Fabio, et mes parents.

« All truths are easy to understand once they are discovered; the point is to discover them » - Galileo Galilei [17]

Remerciements

Je voudrais particulièrement remercier mon maître de stage, le professeur Laurent Schumacher, pour son encadrement et sa patience durant la réalisation de ce mémoire.

Je tiens aussi à remercier tous les membres de ma famille pour l'aide, le soutien, les conseils et la patience dont ils ont fait preuve tout au long de ce travail, sans quoi la réalisation de celui-ci aurait été impossible. Il est effectivement l'aboutissement d'un cycle d'études, qui impliqué les membres de ma famille. Le résultat de travail, qui est la concrétisation de ces différentes années d'études, est également le fruit de ma compagne, qui a su gérer durant les soirées de cours, et l'écriture de ce mémoire, mes 2 petits garçons Matteo et Fabio. Une pensée particulière va à mon père.

Je remercie également tous mes amis, pour leur appui qu'ils m'ont apporté ainsi que pour la relecture de ce document.

Finalement, je tiens également à remercier tout le corps enseignant de l'UNamur, pour le travail qu'il effectue au quotidien ainsi que le temps consacré afin de nous permettre d'avancer tout au long de ce cursus, et ceci dans les meilleures conditions.

Résumé

Les réseaux mobiles n'ont sans cesse continué d'évoluer au fil de ces années pour se transformer de la première norme de réseau mobile (1G) permettant la voix sur un réseau de type circuit, vers un réseau complètement IP (4G) et autorisant des volumes et des débits de données à des vitesses très importantes et très rapides. La prochaine grosse révolution, est ce que l'on nomme la 5G, cinquième génération de standards de réseau mobile.

Cette dernière génération impliquera d'énormes changements dans sa mise en place, que ce soit au niveau de l'infrastructure, de son architecture, et des exigences requises de part et d'autre.

Au fil de ce document, nous aborderons les évolutions des quatre premières normes de réseau mobile avant de nous focaliser sur l'analyse de l'architecture 5G et ses différents éléments. Par la suite, la latence, autre composante de la 5G, sera discutée afin de présenter les éléments qui la composent.

Dans la continuité de la 5G et de ce document, l'aspect de l'agrégation, l'utilisation des multi-canaux seront vus au travers du protocole Multipath TCP (MP-TCP) ainsi que des différents modes de continuité et de session de service (SSC). Ces deux derniers points nous servirons à aborder la suite de notre exposé consacré à la partie recherche.

En effet, l'objectif de cette dernière partie portera sur le développement d'une solution en version couche réseau du mode de continuité et de session de type 2. L'ensemble des points essentiels à la réalisation seront également présentés et développés. Nous terminerons finalement par la présentation des résultats, les enseignements tirés tout au long de ce travail et des problèmes rencontrés.

Abstract

Mobile networks have continued to evolve over the years to transform from the first mobile network standard (1G) for voice over a circuit-like network to a fully IP (4G) network that allows volumes and datas rates at a very fast speeds. The next big revolution is what we call 5G, the fifth generation of mobile network standards.

This latest generation will involve enormous changes in its implementation, whether in terms of infrastructure, architecture, and requirements on both sides.

Throughout this document, we will discuss the evolution of the first four mobile network standards, before focusing on the analysis of the 5G architecture and its various elements. Subsequently, latency, another component of 5G, will be discussed to present the elements that compose it.

In the continuity of the 5G and this document, the aspect of the aggregation, the use of multi-channels will be seen through the TCP Multipath protocol (MP-TCP) and the different ways of continuity and session of service (SSC). These last two points will be used to address the rest of our presentation on the research part.

Indeed, the objective of this last part will be focused on the development of a solution in network layer version of the mode of continuity and session of type 2. All the essential points to the realization will also be presented and developed. Finally we will end with the presentation of the results, the lessons learned throughout this work and the problems encountered.

Table des matières

Dédicaces **3**

Remerciements **4**

Résumé **5**

Abstract **5**

Table des matières **6**

Liste des tableaux **9**

Table des figures **10**

Glossaire **11**

I Etat de l'art **15**

1 Introduction **16**
1.1 Plan 16
1.2 Introduction 17

2 Évolution des architectures mobiles 1G à 4G **18**
2.1 Introduction 18
2.2 CUPS 19

3 Architecture 5G **21**
3.1 Introduction 21
3.2 Objectifs et principes clés 23
3.2.1 Les 8 facteurs clés 23
3.3 Les nouveaux objets connectés (M2M – V2X – VR gaming - 4K vidéo) 25
3.4 Network Slicing 26
3.5 Les usages 28
3.5.1 mMTC 29
3.5.2 eMBB 29
3.5.3 uRLLC 29
3.6 Cloud-Radio Acces Network (CloudRAN ou C-RAN) 29
3.7 Architecture 5G 30

4 La latence **33**

4.1 Les contraintes . . . 33
4.2 Source de la latence . . . 34
4.3 Les temps de transmission . . . 34
4.3.1 Temps total (T) . . . 35
4.3.2 Temps radio . . . 35

5 Multipath TCP **37**
5.1 Introduction . . . 37
5.2 Avantages de MPTCP . . . 38
5.3 Architecture . . . 38
5.4 Connexion MPTCP et le three-way-handshake . . . 39
5.4.1 Handshake des sous flux supplémentaires . . . 39
5.5 Mode de fonctionnement de MPTCP . . . 40

6 Continuité de service et de session (SSC) **41**
6.1 SSC et la 5G . . . 41
6.2 SSC Mode 1 . . . 42
6.3 SSC Mode 2 . . . 42
6.4 SSC Mode 3 . . . 43

II Travail de recherche **44**

7 SSC mode 2 version couche réseau **45**
7.1 Objectifs . . . 45
7.2 Contexte . . . 45
7.3 Méthodologie de réalisation . . . 45
7.4 Proof of Concept . . . 46
7.4.1 Scénario de fonctionnement . . . 46
7.4.2 Architecture réseau . . . 47
7.4.3 Protocole IPv6 . . . 48
7.4.4 MiNinet - Outils de simulation réseau . . . 50
7.4.5 Scapy - Outils de manipulation de paquets . . . 50
7.4.6 Router Advertissement . . . 51
7.4.7 Protocole OpenFlow . . . 53
7.4.8 Connectivité Internet . . . 53
7.5 Exécution et résultats . . . 54
7.5.1 Exécution . . . 55

8 Conclusion **62**
8.1 Enseignement . . . 62
8.2 Problèmes rencontrés (problèmes des ra, freeze, . . .) . . . 62
8.2.1 RA . . . 62
8.2.2 Transit . . . 62
8.2.3 Freeze . . . 63
8.3 Future works . . . 63
8.3.1 Aspects sécuritaires . . . 63

Bibliographie **64**

Code Source **68**

Configuration du serveur Router Advertisement Daemon 68
Code source PYTHON 70
Code source JAVA 101

Liste des tableaux

3.1 Comparaisons entre les performances de la 4G et de la 5G 24

4.1 Latence nécessaire pour les services critiques 33
4.2 Exemple de valeur de latence attendue par une architecture candidate pour la 5G . 34

Table des figures

2.1 Evolution des architectures mobiles au fil du temps . . . 18
2.2 Architecture LTE (Long Term Evolution) . . . 19
2.3 Changement et amélioration de l'architecture comparée à LTE . . . 20

3.1 Planning de déploiement des releases 3GPP . . . 22
3.2 Les 8 facteurs clés en 5G . . . 24
3.3 Cas d'utilisation 5G dans les usines du futur . . . 26
3.4 5G - Tranche réseaux . . . 27
3.5 5G Network Slicing . . . 28
3.6 Architecture physique . . . 30
3.7 Architecture 5G par point de référence . . . 30
3.8 Architecture 5G basé sur le service . . . 31
3.9 Architecture 5G basée sur le service . . . 32

4.1 Architecture LTE - Temps total . . . 36

5.1 Intégration de MPTCP dans la couche transport . . . 38
5.2 Intégration de MPTCP dans la couche transport . . . 39

6.1 Différents modes de continuité des sessions et services . . . 42
6.2 Diagramme de séquence SSC Mode 2 . . . 43

7.1 Architecture réseau IPv6 . . . 47
7.2 Adressage IPv6 . . . 48
7.3 Comparaison de structure entre les entêtes IPv4 et IPv6 . . . 48
7.4 Ordre de chaînage des entêtes IPv6 . . . 50
7.5 Paquet IPv6 créé à l'aide de Scapy . . . 51
7.6 Mécanisme d'auto-configuration SLAAC . . . 53
7.7 Réseau mobile IPv6 Telenet . . . 54
7.8 Interface b-eth0 lors de l'initialisation de la solution . . . 55
7.9 Envoi de paquet Router Solicitation . . . 56
7.10 Réception de paquet Router Advertisement . . . 56
7.11 Configuration diffusée par le serveur RADVD . . . 57
7.12 Calcule d'une adresse IPv6 en EUI-64 . . . 57
7.13 Interface b-eth0 avec l'adresse IPv6 configurée . . . 58
7.14 Basculement du trafic du point d'ancrage 2 à 1 . . . 58
7.15 Paquet IPv6 créé à l'aide de Scapy . . . 59
7.16 État du trafic transitant par l'interface physique et vers Internet . . . 59
7.17 Basculement du terminal d'un point d'ancrage à l'autre . . . 60
7.18 Basculement du trafic du point d'ancrage 3 à 2 . . . 60
7.19 Basculement du trafic du point d'ancrage 3 à 2 . . . 61

Glossaire

Nombres | A | B | C | D | E | F | G | H | I | M | N | O | P | Q | R | S | T | U

Nombres

2G GERAN
3G UTRAN
5G NGCN ou Next Generation Core Network
5GC 5G Core Network

A

AF Application Function
AMF Core Access and Mobility Management Function
AN Access Network
ARQ Automatic Repeat Request
ASF Authentification Server Function

B

Backhaul network Réseau d'amenée
Beamforming Filtrage spatial, formation de faisceaux

C

CAPEX CAPital EXpenditure
CN Core Network
CP Cyclic Prefix
C-RAN Cloud-Radio Acces Network
CRC Cyclic Redundancy Codes
CUPS Control User Plane Separation

D

D2D Device to device
DAD Duplicate Address Detection
DN Data Network

E

E2E End to end
EID Endpoint ID
EIR Equipment Identity Register
eMBB Enhanced Mobile Broadband

EPC Evolved Packet Core
EPS Evolved Packet System
ETE End-to-End
ETR Egress Tunnel Router
EUI-64 "Extended Unique Identifier" ou "identifiant unique étendu"
evolved Node B Station de base des réseaux mobiles basés sur les technologies LTE

F

FDC The Flat Distributed Cloud
FDD Frequency Division Duplexing
FEC Forward Error Correction
FFT Fast Fourier Transform

G

GTP GPRS Tunneling protocol

H

HARQ Hybrid Automatic Repeat Request

I

IETF Internet Engineering Task Force
ILA Identifier Locator Addressing
ILAMP ILA Mapping Protocol
IMEI International Mobile Equipment Identity
IMT-2020 International Mobile Telecommunications-2020
IoT Internet of Things ou Internet des objets
ISI Interférence Inter Symbole
ITR Ingress Tunnel Router

M

MBB Mobile BroadBand ou réseaux mobiles à large bande
MBS Macro cell base stations
MCC Mission Critical Communications
MEC Mobile Edge Computing
MIMO Multiple-Input Multiple-Output
MME Mobility Management Entity
mMTC Massive Machine Type Communications
MPTCP Multipath TCP

N

NEF Network Exposure
Network Slicing Découpe du réseau en tranche virtuelle
NFV Network Function Virtualization
NOMA Non-orthogonal Multiple Access
NR New Radio
NRF NF Repository Function
NRT Non-real time

NSSF Network Slice Selection Functionk

O

OFDM Generalized Frequency Division Multiplexing
OFDMA Orthogonal Frequency Division Multiple Access
OPEX OPerational EXpenditure

P

PAPR Peak-to-Average Power Ratio
PCF Policy Control Function
PDN Packet Data Network
PDU Protocol Data Unit ou Unité de données de protocole
PETR Proxy ETR
P-GW Packet Gateway
PITR Proxy ITR
POTN Packet optical Transport Network
PSNR Peak Signal to Noise Ratio

Q

QAM Quadrature Amplitude Modulation
QoE Quality of Experience
QoS Quality of Service

R

RA Router Advertisement ou Routeur d'annonce
RADVD Router Advertisement Daemon
RAN Radio Access Networks ou réseau d'accès radio
RB Resource block
RFC Requests for comments
RLOC Routing Locator
RRC Resource Radio Control
RS Router Solicitation
RTT Round Trip Time

S

SBS Small cell base stations
SC-FDMA Single Carrier Frequency Division Multiple Access
SDN Software Defined Network ou réseau défini par logiciel
SDR Software Defined Radio
S-GW Serving Gateway
SID Segment Identifier ou identifiant de segment
SMF Session Management Function
SMSF Short Message Service Function
SRH Segment Routing Header
SRv6 Segment Routing IPv6
SSC Session and Service Continuity
SST Standard Slice Type

SVM Support Vector Machine

T

TDD Time Division Duplexing
TTI Time Transmission Interval

U

UDM Unified Data Management
UDR Unified Data Repository
UE User Equipment
UP User Plane ou plan d'utilisateur
UPF User Plane Function
uRLLC Ultra-reliable and Low Latency Communications

Première partie

Etat de l'art

Chapitre 1

Introduction

1.1 Plan

Ce document est divisé en 3 grandes parties qui seront détaillées dans la suite de ce chapitre. Cette découpe correspond aux différentes étapes d'analyse de notre travail de recherche. Nous retrouvons dans la première partie, intitulée état de l'art, la présentation de l'évolution de la technologie mobile au fil du temps. Le chapitre qui suit, est lié à une présentation complète de l'architecture 5G, du protocole MP-TCP et des différentes mode de continuité de session et service. La dernière sera consacrée à notre de travail de recherche.

A la lecture de ce document par le lecteur, celui-ci débutera par un rapide passage en revue des éléments qui ont fait et qui ont permis de faire évoluer les différentes normes de réseaux mobiles pour arriver finalement à la norme actuelle que nous connaissons actuellement, qu'est la 4G.

Après avoir effectué un rappel des changements importants ayant eu lieu au cours des différentes releases, nous aborderons le chapitre suivant dédié à la futur norme de réseau mobile 5G. C'est au travers de ce chapitre que nous en profiterons pour exposer une présentation générale de ce que nous pouvons espérer de cette norme avec également les objectifs attendus, la vision et son architecture. Nous continuerons notre exposé en présentant les différentes sources de la latence dans un réseau afin de permettre au lecteur d'avoir une vision sur les éléments ayant un impact sur celle-ci. Dans le cadre de la compréhension de ce travail, ce chapitre représente un élément destiné à apporter un éclairage dans la compréhension du temps pris lors d'un échange E2E. Cependant, ce chapitre n'étant pas essentielle à la compréhension des points abordés dans la partie recherche, celle-ci peut être éventuellement passé.

Les deux chapitres qui suivent seront dans la continuité de la 5G. En effet la norme 5G autorisera l'utilisation simultanée des connexions Wifi et données mobiles. C'est dans ce cadre-là, que l'utilisation du protocole Multipath TCP pourra s'avérer être un candidat utile afin de pouvoir proposer des débits suffisant à tous les utilisateurs. C'est au cours du chapitre 5 qui nous évoquerons les concepts généraux liés à ce protocole avant d'entamer lors du chapitre 6, un autre aspect également important et lié dans une certaine mesure au protocole MP-TCP. Il s'agit d'une présentation des différentes modes de continuité de session et service possibles lors de l'établissement à un point d'ancrage d'un UE.

Finalement, nous entamerons notre dernière partie et également principale ayant trait à la réalisation d'une version en mode couche réseau du mode de continuité de session et service de type 2. Nous conclurons ce document en présentant par les enseignements acquis au cours de celui-ci

ainsi que par les problèmes rencontrés et les *futurs works* possibles.

1.2 Introduction

Au cours de ces dernières années, les réseaux mobiles n'ont sans cesse continué de se moderniser tant au niveau de l'architecture et que par les services offerts aux utilisateurs. La demande en débit est continuellement en croissance avec l'apparition de nouveaux acteurs, de nouveaux types d'usages, que ce soit par le nombre d'utilisateurs qu'en besoin de bande passante.

Actuellement, LTE permet déjà des vitesses élevées. Cependant, le passage à la 5G impliquera des contraintes encore plus élevées. Des vitesses de connections pouvant atteindre jusqu'à 10 Gbps, des délais de latence de 1ms, un facteur d'augmentation des appareils connectés de l'ordre de 10 à 100x, une perception de la disponibilité de 99.999%, une perception de couverture de 100%, une réduction de la consommation électrique du réseau de l'ordre de 90%, et finalement une autonomie de la batterie jusqu'à dix ans pour les appareils de faible puissance.

De plus, les besoins se diversifient et l'on assiste à l'apparition de nouveaux consommateurs tels que les objets connectés, les communications entre véhicules, la diffusion de vidéo en ultra haute définition (4K), la réalité augmentée, des services critiques. Les vitesses promises pour la nouvelle génération de réseaux mobiles 5G, ont commencé à être commercialisées au cours de cette année au grand public comme c'est déjà le cas avec les opérateurs de téléphonie mobile Swisscomm (Suisse), SK Telecom, KT, LG U (Corée du Sud) et Verizon (Etats-Unis).

Néanmoins, afin de pouvoir offrir ces services, cela implique une refonte importante et une amélioration à différents niveaux de l'architecture. Les chercheurs et ingénieurs étudient et élaborent de nouvelles techniques, de nouvelles interfaces radio, de nouveaux types de codages, modulation, etc.

Dans ce travail, nous partons du point de vue que le lecteur dispose des connaissances de base relatives à l'architecture des normes mobiles précédentes à la 5G et nécessaires à la compréhension de ce document.

L'objectif que nous tenterons d'atteindre, sera d'exposer un inventaire de ce qui pourra être mis en place au cours du déploiement de la 5G dans la première, avant de proposer le développement, la mise en place et les résultats de la transposition du mode de continuité de session et de service de type 2, concept cité à la section précédente.

Chapitre 2

Évolution des architectures mobiles 1G à 4G

2.1 Introduction

Les différentes évolutions des réseaux mobiles ont vu apparaître le service voix, puis la messagerie, l'internet mobile et finalement l'utilisation des applications mobiles multi-services.
Sur un an, le volume de données a presque doublé. Dans 5 ans, celui-ci sera multiplié par 10. Cela nécessite de pouvoir répondre à la demande et d'optimiser l'utilisation des ressources de manière efficace. Nous présentons dans le tableau qui suit les évolutions des architectures cellulaires au fil du temps, que ce soit en termes de débits, de type de technologies ou autres.

La norme LTE fut considérée dans les premiers temps comme une norme de troisième génération (3.9 G) car ne répondant pas entièrement aux normes de 4ème génération. Elle fut spécifiée dans les releases 8 et 9 comme Pre-4G, et fut orientée comme une convergence du tout IP.

Generation→ Features↓	1G	2G	3G	4G	5G
Deployment	1970 – 1980	1990 - 2001	2001-2010	2011	2015-20 onwards
Data Rates	2kbps	14.4-64kbps	2Mbps	200 Mbps to 1 Gbps	1Gbps and higher
Technology	Analog Cellular Technology	Digital Cellular Technology: Digital narrow band circuit data Packet data	Digital Broadband Packet data: CDMA 2000 EVDO UMTS EDGE	Digital Broadband Packet data: WiMax LTE Wi-Fi	wwww Unified IP seamless combination of broadband LAN PAN MAN WLAN
Service	Analog voice service No data service	Digital voice with higher clarity SMS, MMS Higher capacity packetized data	Enhanced audio video streaming video conferencing support Web browsing at higher speeds IPTV support	Enhanced audio, video streaming IP telephony HD mobile TV	Dynamic Information access, Wearable devices with AI Capabilities
Multiplexing Switching	FDMA	TDMA, CDMA	CDMA	CDMA	CDMA
Core Network	PSTN	PSTN	Packet N/W	Internet	Internet
Standards	MTS AMTS IMTS	2G:GSM 2.5:GPRS 2.75:EDGE	IMT-2000 3.5G-HSDPA 3.75G:HSUPA	Single unified standard LTE, WiMAX	Single unified standard
WEB Standard		www	www(IPv4)	www (IPv4)	wwww (IPv6)
Handoff	Horizontal only	Horizontal only	Horizontal & Vertical	Horizontal & Vertical	Horizontal & Vertical
Shortfalls	Low capacity, Unreliable handoff, Poor voice links, Less secure	Digital signals were reliant on location & proximity, required strong digital signals to help mobile phones	Need to accommodate higher network capacity	Being deployed	Yet to be implemented

FIGURE 2.1 – Evolution des architectures mobiles au fil du temps reproduit à partir de *5G Network a New Look into the Future: Beyond all Generation Networks*, par SAHOO, HOTA et BARIK [36]. (jan. 2014) Consulté sur http://pubs.sciepub.com/ajss/2/4/5/index.html

La release 10 a proposé un standard de 4G qui supporte une architecture simplifiée complètement IP grâce au dispositif Mobility Management Entity (MME). Ce dispositif est responsable de l'authentification et du handover. Le handover est une opération permettant à un utilisateur de pouvoir se déplacer d'une antenne à l'autre de manière transparente et tout en continuant de poursuivre son activité sur le réseau. Lorsque le mobile se déplace, le rattachement au nouveau point d'ancrage s'effectue par un transfert via l'interface X2 reliant l'eNodeB d'origine et celle de destination.

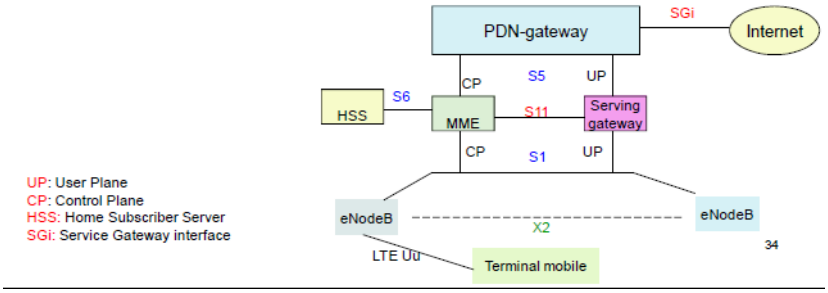


FIGURE 2.2 – Architecture LTE (Long Term Evolution) reproduit à partir de *4G LTE (Long Term Evolution)*, par WEI [41]. (2011) Consulté sur http://www.academia.edu/5964059/4G_LTE_Long_Term_Evolution

Le réseau LTE, qui est actuellement le réseau de dernière génération, est composé de l'Evolved Packet System (EPS).

EPS est en fait composé de deux parties :
- la première partie étant ce que l'on appelle le réseau d'accès (LTE)

- la deuxième partie, le cœur du réseau, qui porte le nom d'ePC (Evolved Packet Core). EPS est un réseau à haut débit et de type IP pour ce qui est du transport de la voix et de données.

Les releases 12-13 furent conçues dans le but d'ouvrir la voie à la pré-5G. Les releases 10-12 correspondent à ce que l'on appelle la LTE-Advanced. A partir de la release 13, on évoque le terme de LTE-Advanced Pro.

2.2 CUPS

Un des changements de la release 14 du 3GPP, fut l'introduction du Control User Plane Separation (CUPS) mettant en œuvre la séparation du plan utilisateur du plan de contrôle des nœuds Evolved Packet Core (EPC).

CUPS a notamment permis la réduction de latence sur le service d'application, par exemple en sélectionnant des nœuds du plan d'utilisateur qui sont plus proches du Radio Access Networks ou réseau d'accès radio ou plus appropriés pour le type d'utilisation de l'équipement d'utilisateur prévu, sans augmenter le nombre de nœuds de plan de contrôle.

Avant CUPS, pour rappel, lorsqu'un UE s'attache au le réseau mobile, celui-ci se connecte via l'eNodeB qui est chargé de la gestion de l'aspect Radio. L'eNodeB est lui-même connecté à ce que l'on appelle le Serving Gateway (S-GW) servant de point d'ancrage local à l'UE. Lorsque

l'UE effectue un déplacement au sein d'eNodeB faisant partie d'une même région, le S-GW sert de point d'ancrage de mobilité. Son autre rôle est la gestion du plan de mobilité de l'utilisateur et l'acheminement des données reçues entre l'eNodeB et le P-GW. Le Packet Gateway (P-GW) servant quant à lui, de passerelle Internet.

Le principe de CUPS est de séparer le S-GW (passerelle régionale) et le P-GW (passerelle d'accès) en deux parties. Le contrôleur en S-GW-C et P-GW-C et le plan de données en S-GW-U et P-GW-U.

CUPS a également pris en compte l'augmentation du trafic en ajoutant des nœuds du plan d'utilisateur sans augmenter le nombre de SGW-C, PGW-C et TDF-C dans le réseau.

Après intégration du CUPS dans l'architecture LTE, les différents éléments (PGW, SGW, MME, HSS, ...) de cette architecture seront intégrés et regroupés dans des fonctions réseau dans l'architecture 5G. Le concept de fonctions réseaux sera en effet présent dans la 5G. Ce point sera abordé plus en détail dans le chapitre suivant.

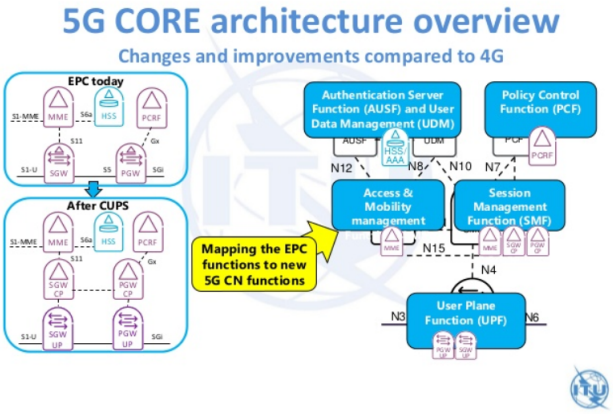


FIGURE 2.3 – Changement et amélioration de l'architecture comparée à LTE reproduit à partir de *5G Network Architecture and FMC - ITU: Committed*, par WILKE [42]. (juin 2017) Consulté sur https://www.itu.int/en/ITU-T/Workshops-and-Seminars/201707/Documents/Joe-Wilke-%205G%20Network%20Architecture%20and%20FMC.pdf

Chapitre 3

Architecture 5G

3.1 Introduction

La prochaine génération de communication mobile ne s'intéressera pas simplement au monde des télécommunications grand public mais également à d'autres domaines tels que le monde des objets connectés, comme les communications entre véhicules par exemple. Cela impliquera de nouvelles spécifications que ce soit par une latence très faible de bout en bout, ou par une fiabilité et une robustesse du réseau.

En effet, ces nouveaux usages auront des besoins spécifiques. L'architecture réseau LTE actuelle ne permet pas de satisfaire tous ces nouveaux types d'usages et leurs exigences futures, qu'en augmentant simplement le nombre d'équipements. L'architecture de réseau mobile existante a été conçue pour répondre aux exigences des services voix et MBB (Mobile BroadBand ou réseaux mobiles à large bande) classiques. Cela impliquera donc des changements drastiques au sein de l'architecture réseau actuelle. L'un des aspects sera d'améliorer le délai au sein du RAN et du cœur du réseau, mais également au niveau du backhaul.

Cette nouvelle norme est spécifiée principalement par deux acteurs importants :

- l'UIT (Union internationale des télécommunications)
- et le 3GPP (3rd Generation Partnership Project)

L'UIT, l'agence des Nations Unies spécialisée dans les technologies de l'information et de la communication, réalise des études par l'intermédiaire de son « Working Party 5D », son sous-groupe en charge de traiter les questions techniques et d'exploitation relatives aux radiocommunications.

De l'autre côté, le 3GPP a été instauré en 1998 et regroupe sept organismes de standardisation, plusieurs centaines d'industriels, des associations et des organismes publics. Les deux rôles liés au 3GPP sont le développement et la maintenance des spécifications techniques relatives aux normes de téléphonie mobile.
Lors de la spécification d'un nouveau standard par l'UIT, le 3GPP travaille, lui, sur les solutions techniques qui permettent de répondre aux objectifs.

La première rédaction de la norme relative à la 5G sera la release 15, qui est en cours d'élaboration. Celle-ci a débuté en décembre 2016 avec l'étude d'une nouvelle architecture. En mars 2017 a débuté l'étude d'une nouvelle interface air (appelé NR pour New Radio). La première version

a été approuvée et formalisée durant le courant de la fin d'année 2018. Une seconde release, la release 16, est quant à elle attendue pour mars 2020, soit 18 mois entre chaque release.

La mise en place de la 5G s'est déroulée en 2 phases :

- **Phase 1 :** Phase de spécification eMBB.
- **Phase 2 :** Reste de la spécification des technologies uRLLC et mMTC.

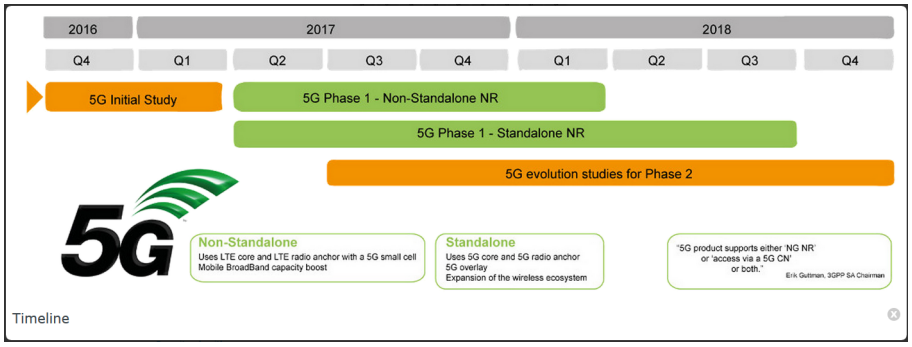


FIGURE 3.1 – Planning de déploiement des releases 3GPP reproduit à partir de *Mobile, LTE, 5G, building blocks, MEC, multi-access Edge Computing, NFV Network Functions Virtualisation, NGP, mWT*, par DAHMEN-LHUISSIER [7]. (nov. 2018) Consulté sur https://www.etsi.org/technologies/5g

La transition entre l'architecture LTE et la 5G, se fera de manière progressive. En effet, LTE continuera d'évoluer en parallèle à la NR. Ces deux normes continueront de coexister et seront dans un premier temps, complémentaires.

Dans les premiers temps du déploiement de la 5G, LTE restera maître du réseau et contrôlera les antennes NR. De plus, certains objectifs de la 5G pourront être atteints avec des fonctionnalités et technologies mises en place avec les releases 13, 14 et 15. Cela représentera plutôt une évolution de la LTE. On parlera éventuellement de LTE Advanced Pro ou 4.9G.

La LTE Advanced Pro aura recours notamment à l'utilisation du massive MIMO (Multiple-Input Multiple-Output), l'agrégation des porteuses (carrier agregation), ou au NFV (Network Function Virtualization). L'agrégation des porteuses consiste en un regroupement de plusieurs bandes fréquences, attribuées à un utilisateur, et dont le but final est de permettre des transferts à des vitesses plus rapides.

D'autres technologies seront quant à elles mises en place dès que les pré-requis liés à celles-ci seront remplis. On peut notamment citer les porteuses NR en bandes millimétriques, le NOMA (non orthogonal multiple access) ou le MEC (mobile edge computing).

Cette architecture devra supporter un large spectre de fréquences, celles en dessous des 6 Ghz plus appropriées pour prendre en charge les services tels que le mMTC où la couverture est la plus importante. Pour les fréquences supérieures au 6 Ghz, elle sera indispensable pour assurer la capacité maximale demandée par les services eMBB. La 5G devra supporter également un ensemble d'interfaces radio intégrées, le transfert transparent entre technologies d'accès radio.

La prochaine norme 5G sera une technologie de rupture par rapport aux architectures précédentes. Elle autorisera une expérience utilisateur qui sera toujours adaptée à l'usage courant. 5G ne sera pas non plus une technologie universelle mais polymorphe, qui devra s'adapter aux différents besoins. Étant donné qu'il est impossible de répondre à tous les besoins en même temps, il est nécessaire de découper les différents usages en tranches ou Network Slicing, car comme nous le verrons plus tard, chaque tranche permettra d'atteindre un ou plusieurs des huit facteurs clés définissant les objectifs attendus de la 5G.

Le réseau sera également structuré de manière à toujours utiliser la meilleure ressource disponible pour chaque nouvelle demande d'utilisation applicable au contexte de l'utilisateur. Cette nouvelle architecture sera basée sur un Cloud dynamique qui sépare le plan utilisateur (User Plane) et le plan de contrôle (Control Plane) et elle sera donc plus "plate". On parlera notamment d'architecture plate et distribuée (FDC ou The Flat Distributed Cloud).

La majorité des fonctions du cœur du réseau seront déployées en 5G en tant que fonctions de réseau virtuelles (VNF), s'exécutant ainsi dans des machines virtuelles sur des serveurs standard et potentiellement sur des infrastructures de cloud computing, c'est-à-dire dans des centres de données.

La conception de ces fonctions exploitera les principes du réseau défini par logiciel (SDN), tels que la division du plan utilisateur et du plan de contrôle.

Le système 5G supportera toutes les capacités EPS. Néanmoins certaines restrictions seront présentes avec la 5G et les nomes précédentes. Nous pouvons citer notamment le cas où aucune mobilité ne pourra être possible entre le réseau 5G et les réseaux de type circuit. Le basculement entre un réseau 5G et un réseau de type 2G (GERAN) ou UTRAN (3G) ne seront pas autorisés. Par contre, le handover entre et depuis un réseau de type 4G, 5G et un réseau wifi sera possible. De plus, l'accès depuis un réseau 2G et 3G vers le cœur d'un réseau 5G ne sera pas réalisable. En final, les 3 types d'accès supportés par la 5G seront E-UTRAN, WLAN et la NR (New Radio)

3.2 Objectifs et principes clés

3.2.1 Les 8 facteurs clés

Les objectifs de la 5G seront de permettre la réalisation d'un réseau extrêmement fiable avec des performances plus homogènes, quelle que soit la distance entre un terminal et l'eNodeB, avec une connexion stable, même en mobilité (avec des vitesses en déplacement de l'ordre de 500 km/h), et une augmentation de l'efficacité énergétique (batteries jusqu'à 100 fois moins énergivores).

Comme nous le verrons plus tard, afin de pouvoir réaliser les trois types d'usages prévus en 5G, huit indicateurs de performance (KPI – Key performance indicators) ont été établis par l'UIT pour préciser, quantifier et mesurer les caractéristiques de systèmes IMT-2020 (5G) :

- ▷ Débit crête par utilisateur (Gbit/s) ;
- ▷ Débit moyen perçu par l'utilisateur (Mbit/s) ;
- ▷ Efficacité spectrale (bit/Hz) ;

- ▷ Vitesse maximale des terminaux (km/h) ;
- ▷ Latence (ms) ;
- ▷ Nombre d'objets connectés sur une zone (quantité d'objets/km²) ;
- ▷ Efficacité énergétique du réseau ;
- ▷ Débit sur une zone (Mbit/s/m²) ;

La figure reprise ci-dessous illustre les 8 facteurs clés qui devront être mis en œuvre en 5G en comparaison avec les valeurs actuelles des réseaux 4G/LTE. Il s'agit des exigences minimum requises par l'International Mobile Telecommunications-2020 (IMT-2020).

	Performance	Génération	
		4G	5G
1	Débit maximal (Gbit/s)	10	100
2	Débit aperçu par l'utilisateur (Mbit/s)	1	20
3	Efficacité spectrale	1x	3x
4	Vitesse (km/h)	350	500
5	Latence (ms)	10	1
6	Nombre d'objets connectés sur une zone (quantité d'objets/km²)	10 exp 5	10 exp 6
7	Efficacité énergétique du réseau	1x	100x
8	Débit sur une zone (Mbit/s/m²)	0.1	10

TABLE 3.1 – Comparaisons entre les performances de la 4G et de la 5G

La figure reprise ci-dessous nous présente par contre, les mêmes facteurs clés mais sous forme de graphique.

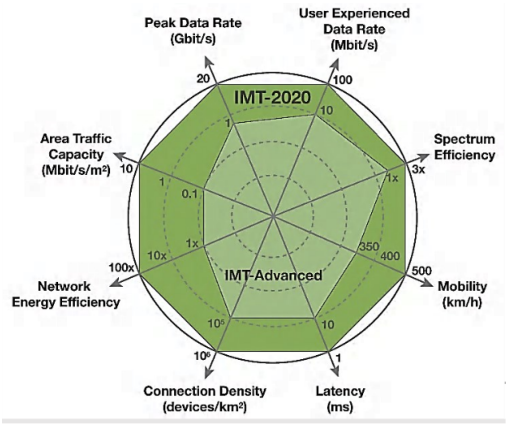


FIGURE 3.2 – Emerging Trends in 5G/IMT2020 reproduit à partir de *Les nouveaux enjeux de la 5G*, par *Les nouveaux enjeux de la 5G* [25]. (mar. 2017) Consulté sur https://www.arcep.fr/uploads/tx_gspublication/rapport-enjeux-5G_mars2017.pdf

Les réseaux 5G devront, sur base de ces 8 huit indicateurs clés de performance, être en mesure de pouvoir fournir des services diversifiés, prendre en charge les accès coexistants de plusieurs

normes (5G, LTE et Wi-Fi) et coordonner différents types de sites (stations de base macro, micro et pico) tout en permettant de :

- Séparer le plan de contrôle et du plan d'utilisateur (CUPS)
- Réduire la dépendance entre le cœur du réseau (CN) et le réseau d'accès (AN)
- Scinder les ressources calculées des ressources de stockage
- Autoriser l'utilisation simultanée de services situés de manière locale et ceux de type centralisés. Pour les services ayant une contrainte de latence très faible ou bien d'accès local, les fonctions du plan d'utilisateur devront être placées au plus proche du réseau d'accès
- Découper les fonctions réseau en modules pour permettre une conception du réseau de manière flexible et efficace

Le défi du design sera donc de pouvoir créer une architecture réseau capable de supporter une telle flexibilité tout en répondant à des demandes d'accès différenciées afin de satisfaire la demande.

3.3 Les nouveaux objets connectés (M2M – V2X – VR gaming - 4K vidéo)

Avec cette nouvelle génération d'architecture, il sera aussi le cas de voir apparaître de nouveaux types d'usages différents. La 5G sera destinée à de nombreux secteurs variés et qui auront un intérêt commun pour cette nouvelle norme communication mobile. En effet, il pourra s'agir par exemple des secteurs de l'énergie, de la santé, de l'industrie, du transport ou bien des médias par exemple.

Le secteur de l'énergie a, par exemple, vu apparaître de nouvelles énergies renouvelables, l'ouverture à la concurrence, et l'arrivée de nouvelles formes de producteurs d'énergie, que ce soient des sociétés indépendantes ou des citoyens. De plus, avec l'augmentation des prix des énergies fossiles, le but est de permettre une gestion plus efficace et plus réactive de ces réseaux.

Dans le secteur de la santé, on pourrait citer par exemple, le cas d'opérations chirurgicales à distance nécessitant des communications avec une latence très faible et une fiabilité très forte. On parlera dans ces cas-là d'applications de missions critiques. (Mission Critical Communications)

Dans le secteur des transports, les véhicules autonomes pourront prendre des décisions sans intervention humaine, mais pourront interagir entre elles (V2V). Cela impliquera des temps de réactions compatibles avec les exigences d'un déplacement à grande vitesse.

Le secteur de l'industrie sera aussi concerné par le changement. On parle notamment de l'industrie 4.0. Certaines utilisations plus pratiques de "l'usine du futur" ont déjà été définies et analysées par 3GPP, avec un certain nombre d'acteurs de l'industrie, dans le rapport technique TR 22.804.

Parmi les cas d'applications industrielles, nous pouvons citer le contrôle des mouvements de la robotique mobile, le monitoring, l'automatisation des processus, l'accès et la maintenance

à distance des équipements. Ces cas d'utilisation et d'applications nécessiteront des débits de données très élevés qui pourront être fournis par l'eMBB.

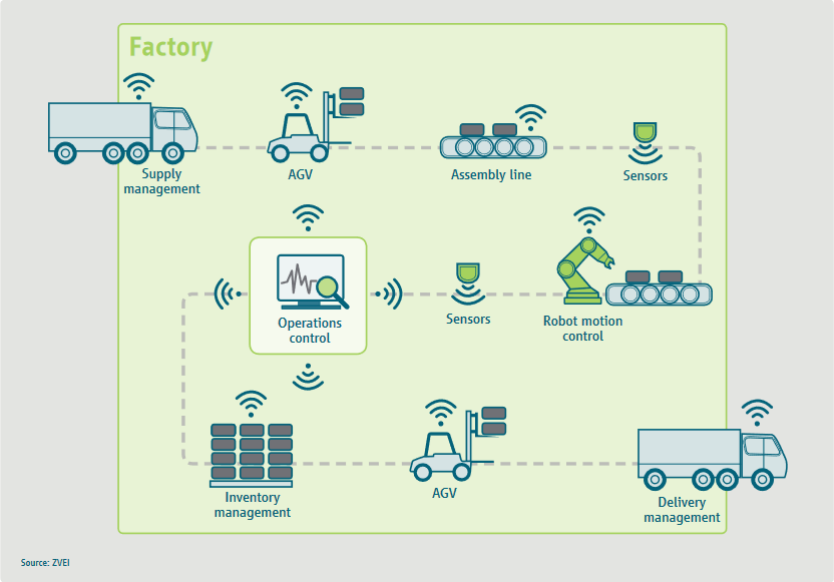


FIGURE 3.3 – Exemple de cas d'utilisation 5G dans le cadre des usines du futur reproduit à partir de *5G for Connected Industries and Automation*, par 5G-ACIA [1]. (nov. 2018) Consulté sur https://www.5g-acia.org/fileadmin/5G-ACIA/Publikationen/Whitepaper_5G_for_Connected_Industries_and_Automation/5G-for-Connected-Industries-and-Automation-White-Paper.pdf

Dans le futur, les transmissions vidéo auront lieu en ultra haute définition (4K), voire en 8K, ce qui nécessitera des débits très importants également.

D'autres technologies seront entre autre possibles grâce à la 5G, comme la réalité augmentée (AR), la réalité virtuelle (VR) et le jeu en temps réel par exemple. Ces technologies s'appuieront aussi sur des débits importants et sur une certaine latence.

3.4 Network Slicing

La réalisation de la 5G passera par une découpe du réseau en tranches afin de garantir les différents services 5G. Ceux-ci seront la clé de l'évolution de l'architecture réseau. Ce découpage sera rendu possible en ayant comme base le NFV (virtualisation des fonctions) et le SDN ("soft-warisation" du réseau).

Concrètement, il s'agit d'une technique consistant à découper une infrastructure physique unifiée en une architecture virtuelle et dont le but est de pouvoir répondre à différents types d'usages.

La figure ci-dessous illustre comment diverses tranches de réseau peuvent être utilisées pour fournir une QoS (Quality of Service) distincte à chaque type de périphérique.

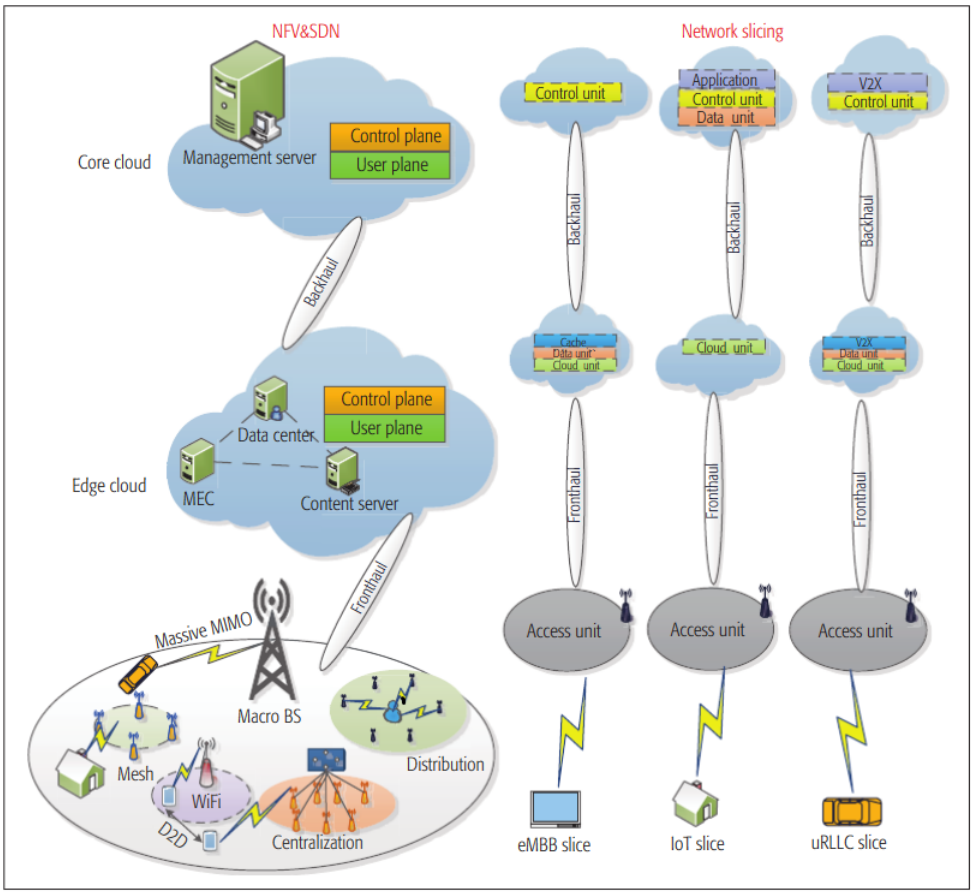


FIGURE 3.4 – 5G - Tranche réseaux reproduit à partir de « Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges », par ZHANG, LIU, CHU, LONG, AGHVAMI et LEUNG [43]. (jan. 2017) Consulté sur https://www.researchgate.net/publication/313611684_Network_Slicing_Based_5G_and_Future_Mobile_Networks_Mobility_Resource_Management_and_Challenges

Le découpage en tranches permettra aux opérateurs de limiter, pour chaque tranche, l'accès aux seules ressources dont celle-ci a réellement besoin.

Cela aura pour avantage, par exemple, de ne pas devoir fournir de la puissance de calcul aux périphériques qui n'en ont pas besoin, par exemple aux IoT (Internet of Things ou Internet des objets). Cela permettra éventuellement de réutiliser des équipements plus anciens. Par ailleurs, l'avantage du découpage en tranches est l'isolation. Chaque tranche dispose de sa propre RAN et chaque tranche réseau sera séparée comme illustré ci-dessus.

La tranche eMBB pourrait être la tranche par défaut attribuée à l'abonné lors de la connexion au réseau.

La tranche des objets connectés (mMTC) aura quant à elle un type de radio différente. Par contre, vu que leur nombre en termes d'abonnés sera important, ils ne requerront qu'un débit très faible.

La tranche URLLC nécessitera quant à elle que le plan utilisateur soit très près de l'abonné afin de réduire la latence à moins d'1 ms.

En ce qui concerne les éléments de l'architecture réseau, le RAN sera important pour achever le partage des ressources radio. Ceci sera réalisé à l'aide de SDN ou C-RAN, concept que nous aborderons par la suite.

Le contrôleur dans le C-RAN allouera des ressources de calcul, de réseau et de radio appropriées selon les diverses tranches du réseau.

L'architecture CUPS pourra être utilisée pour aider, et pour permettre d'affecter les ressources nécessaires des différentes tranches réseau.

Le découpage du réseau de bout en bout inclura également le cœur du réseau (Core Network), le RAN et éventuellement le réseau de transport (backhaul ou réseau d'amenée). Le cœur du réseau sera implémenté à l'aide des fonctions réseau et de SDN. Ces fonctions réseau virtuelles permettront sans problème qu'elles soient adaptées afin de se calquer sur les besoins ainsi que les contraintes des exigences de service et de performances.

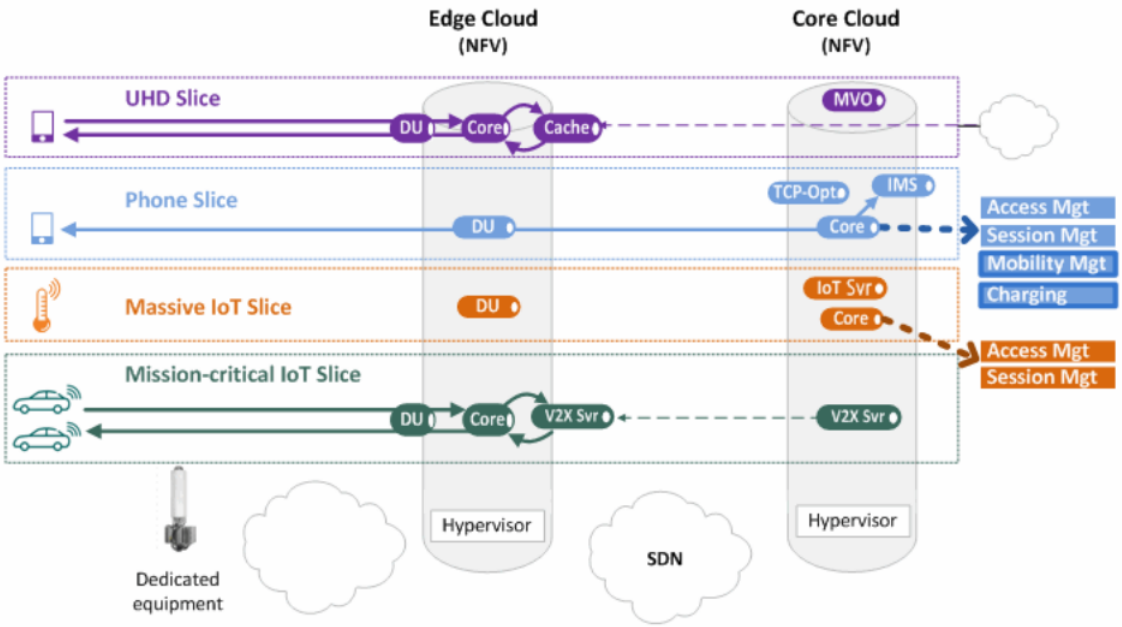


FIGURE 3.5 – 5G Network Slicing reproduit à partir de *E2E Network Slicing - Key 5G technology : What is it? Why do we need it? How do we implement it?*, par SON et YOO [37]. (nov. 2015) Consulté sur https://www.netmanias.com/en/?m=view&id=blog&no=8325

Dans le réseau d'amenée, on pourra compter sur SDN afin de déterminer le meilleur chemin pour un paquet de traverser les différents éléments du réseau. Une des solutions proposées, pourrait être basée sur l'utilisation du Segment Routing IPv6.

3.5 Les usages

Lors de la spécification de cette nouvelle norme, trois grandes catégories d'usages ont été définies par l'UIT, sous le terme IMT- 2020. Or les exigences de ces différentes catégories font

qu'elles sont potentiellement incompatibles les unes aux autres. Ces catégories permettront de répondre aux différents besoins que nous allons décrire ci-après.

Tout d'abord, il y a une première catégorie appelée Massive Machine Type Communications (mMTC) , qui englobe tout ce qui correspond aux communications avec un grand nombre d'objets mais ayant des exigences en terme de qualité de service différentes. La finalité est la capacité à répondre à l'augmentation exponentielle de la quantité d'objets connectés de l'ordre de 10^6.

Une seconde catégorie, est celle de l'eMBB ou Enhanced Mobile Broadband. Cette catégorie comportera les connexions en ultra haut débit et en extérieur avec QoS uniforme en intérieur, et ce, jusqu'à la limite de la cellule.

La dernière catégorie aura trait à ce que l'on nomme l'uRLLC – Ultra-reliable and Low Latency Communications. Cette catégorie aura comme finalité les communications ultra-fiables liées à la fourniture des services critiques, nécessitant une latence très faible ainsi qu'avec une réactivité importante.

3.5.1 mMTC

Ce type de catégorie implique une couverture étendue, une faible consommation énergétique, des débits très faibles, et la capacité à gérer des objets répartis de manière très dense. Le réseau ne fournira par contre pas une utilisation efficace du spectre ou une faible latence. Cette catégorie englobera principalement l'Internet des objets (IoT).

3.5.2 eMBB

Il s'agira du service nécessitant une zone de couverture importante et une connexion toujours plus rapide destinée aux applications et aux services, avec des débits de données plus importants allant jusqu'à plusieurs Gbit/s, et ce, de manière fiable. Dans ce type de couche, l'efficacité spectrale, le débit maximum ainsi que la capacité pourront être réalisés au détriment de la latence ou du nombre de connexions simultanées.

3.5.3 uRLLC

L'ultra-reliable low latency (uRLLC) requerra une latence de bout en bout inférieure à 1 ms ainsi qu'une transmission avec une fiabilité de l'ordre de 99.999 %.

3.6 Cloud-Radio Acces Network (CloudRAN ou C-RAN)

Le Cloud-Radio Acces Network est un terme faisant référence au concept de virtualisation des fonctionnalités de la station de base au moyen du Cloud Computing. L'architecture C-RAN décompose la station de base en deux parties :

- **RRH** : correspond aux têtes radio sur les sites cellulaires ;
- **BBU** : correspond aux unités de traitement de la bande de base mises en commun dans un pool centralisé.

Le lien entre les deux éléments est une interface reliée par une fibre optique. C-RAN a pour objectif d'augmenter la couverture et la capacité du réseau, tout en réduisant les délais de déploiement, la consommation énergétique et les coûts d'investissement et d'exploitation nécessaires pour déployer et maintenir des réseaux hétérogènes denses.

Elle permet également une gestion globale des ressources, une gestion efficace des interférences et une meilleure utilisation de la technologie d'accès radio (RAT).

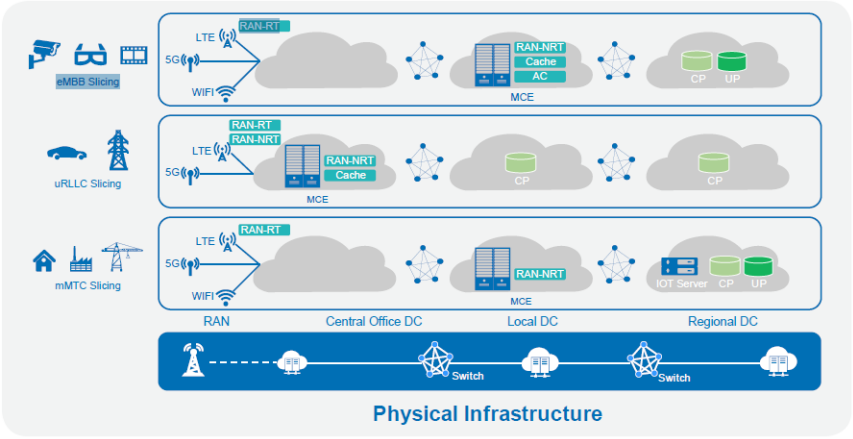


FIGURE 3.6 – 5G Network Architecture - A High-Level Perspective reproduit à partir de *White Paper: 5G Network Architecture - A High-Level Perspective - Industry insight in Huawei*, par HUAWEI [22]. (déc. 2016) Consulté sur https://www.huawei.com/en/industry-insights/outlook/mobile-broadband/insights-reports/5g-network-architecture

3.7 Architecture 5G

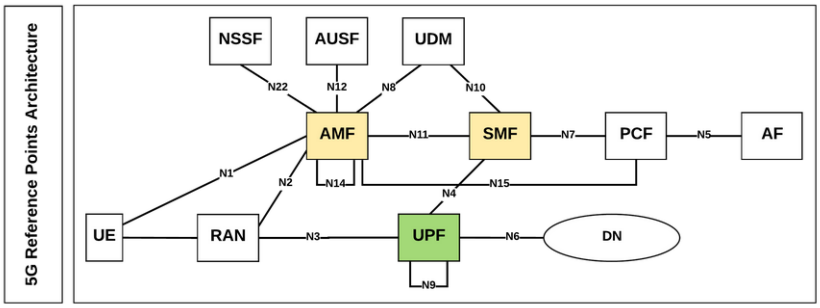


FIGURE 3.7 – Architecture 5G par point de référence reproduit à partir de *Core Network Evolution - 5G Service based Arhcitecture*, par RABIE [34]. (déc. 2017) Consulté sur https://netmanias.com/en/post/blog/12967/5g/core-network-evolution-5g-service-based-arhcitecture

L'architecture 5G sera composée d'une série de fonctions réseau (NF, Network Function) dont chacune sera dédicacée à un type de service précis. Les fonction sont décrites de manière générale et nous renvoyons éventuellement le lecteur qui souhaiterait approfondir ces différentes fonctions vers les sources de la release 15 du 3GPP.

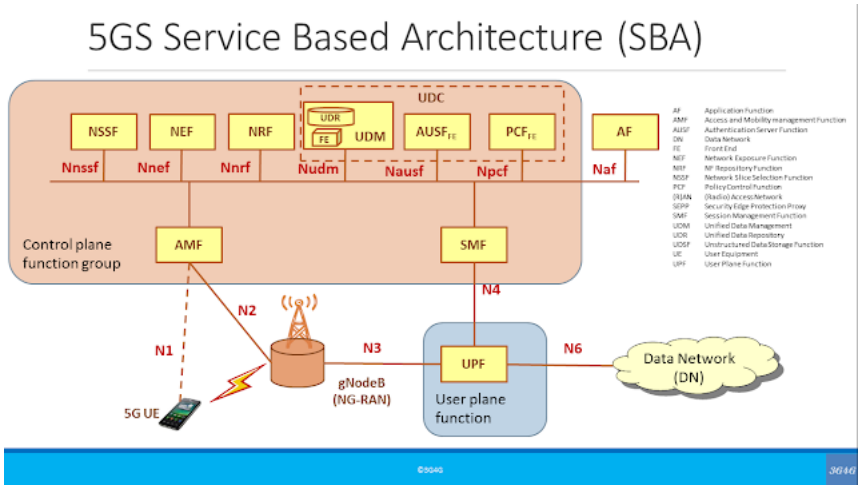


FIGURE 3.8 – Architecture 5G basé sur le service [a]

a. https ://blog.3g4g.co.uk/search/label/5G

- **Authentification Server Function (AUSF)** : traitera de l'authentification de l'équipement
- **Core Access and Mobility Management Function (AMF)** : assurera la gestion de la mobilité de l'UE. Les éléments MME, S-GW-C et P-GW-C présent en LTE ont été scinder entre les fonctions AMF et SMF.
- **Session Management Function (SMF)** : gestion de la session de l'UE
- **Data Network (DN)** : représente l'accès Internet, les services de l'opérateur
- **Network Exposure (NEF)** : permet d'exposer en toute sécurité les services et les capacités réseau
- **Policy Control Function (PCF)** : traite de la politique applicable à l'UE que ce soit en terme de mobilité, de qualité de service que de la sélection de la technologie d'accès
- **Short Message Service Function (SMSF)** : traite de l'envoi et de la réception des SMS de l'UE
- **Unified Data Management (UDM)** : sert d'interface aux fonctions réseau qui doivent accèder aux données de souscription de l'UE
- **Unified Data Repository (UDR)** : consiste en une base de données de l'ensemble des informations relatives à l'utilisateur. Y sont stockées les données de souscription, et la politique de celui-ci notamment
- **User Plane Function (UPF)** : traite du plan de données en entrée et en sortie de l'UE

- **Application Function (AF)** : influence applicative sur le routage du trafic, interaction avec le cadre de politique pour le contrôle des politiques

- **Equipment Identity Register (EIR)** : base de données comportant les informations d'identification (IMEI) et de sécurité d'un terminal. Autorise la vérification de l'identité du terminal afin de s'assurer qu'il est autorisé

- **NF Repository Function (NRF)** : fournit les informations permettant à des fonctions réseau client d'interagir avec une fonction réseau serveur

- **Network Slice Selection Functionk (NSSF)** : permet d'identifier la fonction AMF appropriée pour la prise en charge de la gestion de la mobilité l'UE

Les fonctions réseau mettront à disposition leurs fonctionnalités via des interfaces basées sur le service à l'aide d'API.

Dans le cas de l'architecture basée sur les services, la principale différence se situe dans le fait que le plan de contrôle disposera de fonctions qui interrogeront le NF repository (NRF) pour connaître et dialoguer avec les autres fonctions au lieu d'avoir un nombre d'interfaces prédéfinies. Ceci permet une flexibilité dans la mise en place de ce modèle.

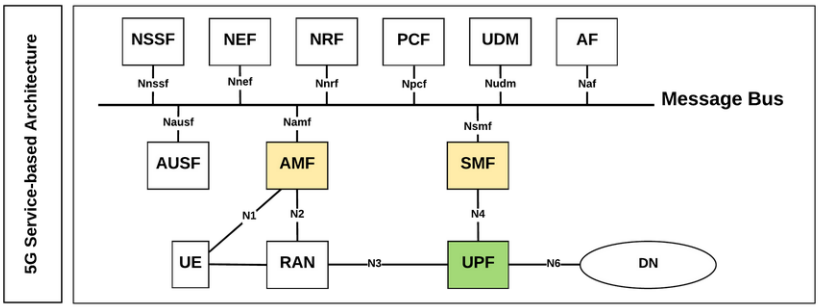


FIGURE 3.9 – Architecture 5G basée sur le service reproduit à partir de *Core Network Evolution - 5G Service based Arhcitecture*, par RABIE [34]. (déc. 2017) Consulté sur https://netmanias.com/en/post/blog/12967/5g/core-network-evolution-5g-service-based-arhcitecture

Chapitre 4

La latence

Dans cette section, nous proposons d'analyser le temps requis pour le traitement et la transmission d'un paquet, que ce soit depuis le terminal de l'utilisateur, par l'eNodeB, et les autres éléments du réseau, pour traverser l'ensemble de l'architecture réseau de bout en bout. Nous passerons en revue les éléments ayant une latence importante et pouvant être améliorés et permettant une réduction de la latence.

En effet, au fil de l'évolution des normes de réseaux mobiles cellulaires, la latence acceptable est passée en 2G entre 500 à 1000 ms, à 200 ms en 3G, à 100 ms en 4G et finalement à 1 ms en 5G.

Il est important de comprendre les éléments du réseau où la latence se situe et ainsi que son importance.

4.1 Les contraintes

La latence de bout en bout (E2E) dans la prochaine norme devra être aussi basse que possible, de l'ordre de 1 ms pour atteindre une disponibilité de 99.99 %, notamment pour les services critiques ayant besoin d'une latence et d'une fiabilité très forte.

La 5G permettra d'offrir de nouvelles possibilités d'utilisation à des secteurs d'activités très variés. Ceux-ci requerront en effet une latence différente afin de pouvoir réaliser leurs finalités.

Le tableau repris ci-dessous illustre quelques cas ainsi que leurs valeurs de latence requise.

Latence requise pour les différents services critiques

Paramètres	Valeur (ms)
Automatisation d'usine	0.25 - 10
Système de transport intelligent	10-100
Téléprésence et robotique	< 1
Réalité virtuelle	< 1
Soins de santé	1-10
Jeux	< 1
Réseau intelligent	1-100

TABLE 4.1 – Latence nécessaire pour les services critiques

4.2 Source de la latence

Dans les réseaux LTE actuels, la source de la latence peut être divisée en 2 parties : la latence vue au plan d'utilisateur et celle liée au plan de contrôle.

Dans le cadre du plan d'utilisateur, celle-ci correspond au temps nécessaire pour transmettre un paquet depuis l'UE à travers le réseau de l'opérateur vers l'Internet et revenir.

Le délai est donc impacté par le parcours au sein du réseau en passant par les éléments suivants : le RAN, le réseau d'amenée (Backhaul network), le cœur du réseau et le réseau de données (Data Network ou DN) ou l'Internet.

Le plan de contrôle, quant à lui, se mesure par le temps de transition nécessaire à l'UE pour passer d'un état inactif à un état actif. Pour un état inactif, cela correspond au fait que l'UE ne soit pas connecté avec le RRC (Resource Radio Control)

4.3 Les temps de transmission

Au cours de ce chapitre, nous nous intéresserons principalement au temps de transmission pour le plan d'utilisateur. Le temps de transmission total peut être vu comme la somme des temps de parcours nécessaires au niveau des différentes parties du réseau qui comprend le temps de propagation, de transmission de traitement et de retransmission.

Le temps total peut donc être exprimé de la façon suivante :

$$T = T_{Radio} + T_{Backhaul} + T_{Core} + T_{Transport}$$

Paramètres	Valeur (ms)
Temps de traitement de l'équipement utilisateur	0.3
Temps de traitement de l'eNB	0.3
Temps minimum pour transmettre	0.2
Temps requis pour la liaison montante	0.2
Temps nécessaire pour le lien montant ACL/NACK Transmission	0.06 (couverture courte) 0.25 (couverture moyenne) 0.50 (couverture élevée) 1.00 (couverture extrême)
Ressources des liaisons descendantes pour les blocs TDD bidirectionnels avec une durée de 1ms	0.80 (DL heavy block) 0.40 (balanced block) 0.20 (UL heavy block)
Ressources des liaisons descendantes pour les blocs TDD bidirectionnels avec une durée de 4ms	2.8 (DL heavy block)

TABLE 4.2 – Exemple de valeur de latence attendue par une architecture candidate pour la 5G

Comme indiqué dans l'étude réalisée par les auteurs de [31], les temps de transmission sont décrits comme suit :

4.3.1 Temps total (T)

- **TRadio** est le temps pour transmettre un paquet entre les evolved Node B et le terminal de l'UE. Il est dû essentiellement à la couche physique. Le temps T_{Radio} devrait, en termes de latence, ne pas dépasser 0.5ms.

  Les améliorations possibles pourraient porter notamment sur la structure des trames, le codage, l'utilisation de nouvelles formes d'onde.

- **TBackhaul**, correspond au temps pour établir la connexion avec le cœur du réseau et l'eNodeB. La connexion peut être établie via fibre optique, paires cuivrées, sans fil.

- **TCore** : il s'agit dans ce cas-ci, du temps de traitement pris par le cœur du réseau.

- **TTransport** : le délai pour la communication de données entre le réseau de base et Internet ou le Cloud.

Le délai de bout en bout (TE2E) peut donc être défini comme approximativement 2 x T.

4.3.2 Temps radio

- **tQo** correspond au délai d'attente et dépend du nombre d'utilisateurs qui seront multiplexés sur les mêmes ressources.

- **tFA** est le délai dû à l'alignement du cadre qui dépend de la structure de trame et les modes de duplexage (c'est-à-dire, la *Frequency Division Duplexing (FDD)* et duplexage par répartition dans le temps (TDD)).
- **tTx** est le temps de traitement de la transmission.

- **tTBsp** est le délai de traitement de transmission de la station de base.

- **tMpt** est le délai de traitement du terminal de l'utilisateur. Le délai dépend de la capacité de la station de base et de celle du terminal de l'utilisateur.

Le facteur tTx est le temps nécessaire pour le traitement de la transmission, et la transmission de la charge utile qui utilise au moins un TTI (Time Transmission Interval) en fonction des conditions du canal radio, de la taille de la charge utile, de la ressources disponible, et des erreurs de transmission et retransmission ;

Temps Core

Lors du déploiement de l'architecture 5G, les technologies telles que le SDN, le NFV et autres approches de réseaux intelligents, comme le FOG/MEC et système de caching, qui permettent de réduire la latence.

Temps Backhaul

Temps Transport MEC / FOG Computing / Cloud / Mise en cache permet de réduire également le délai

A titre d'exemple, dans les réseaux LTE, le temps de transmissions est de l'ordre de 10ms.

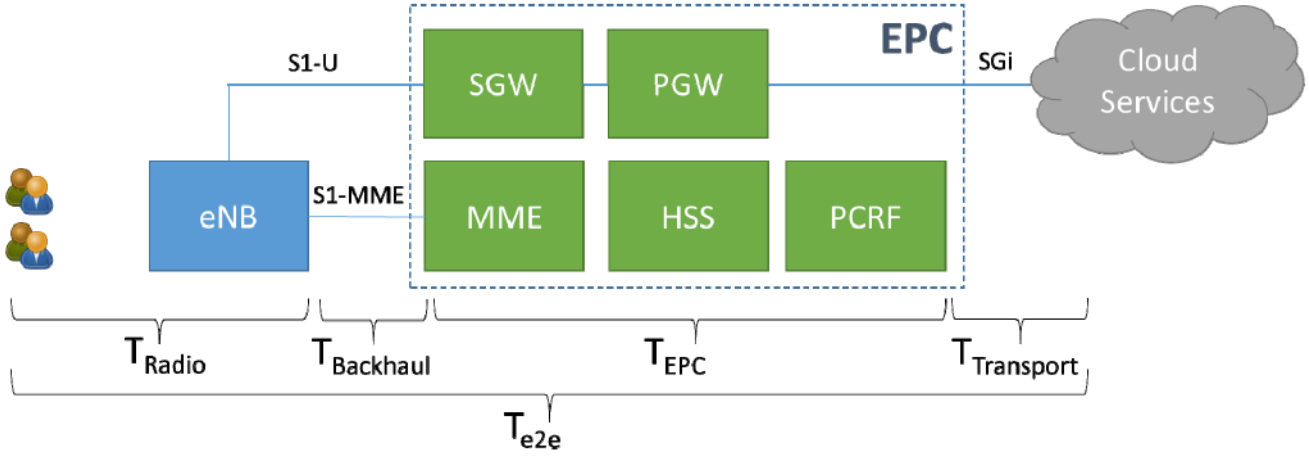


FIGURE 4.1 – Architecture LTE - Temps total reproduit à partir de « Enabling Low Latency Services on LTE Networks », par GARCIA-PEREZ et MERINO [18]. (sept. 2016) Consulté sur https://www.researchgate.net/profile/Cesar_Garcia_Perez/publication/311716854_Enabling_Low_Latency_Services_presentation/links/5857ac3d08ae8f695559f337/Enabling-Low-Latency-Services-presentation.pdf?origin=publication_list

Chapitre 5

Multipath TCP

5.1 Introduction

Actuellement, à l'heure de l'Internet, où les besoins en bande passante ne cessent d'augmenter, où les nœuds permettant le trafic sont de plus en plus nombreux et connectés entre eux, le protocole TCP n'utilise pour l'instant qu'un seul chemin à la fois par connexion. On estime que d'ici 2025, le volume de données sera multiplié par 5.

Avec l'arrivée de la future norme de GSM 5G, l'agrégation et l'utilisation des multicanaux se feront de plus en plus en association avec différents types de supports. Dans le cas d'un équipement mobile, celui-ci aura la capacité d'être connecté notamment via le réseau mobile de son opérateur (3G, 4G, ...) mais de pouvoir également transmettre de manière transparente et via une seconde connexion de données qui pourrait être via le réseau wifi ou tout autre type de connexion. Cependant, la version actuelle de TCP ne tire pas profit de cette situation en ne permettant pas de transmettre pour une même connexion, sur plusieurs interface à la fois.

Une solution permettant de tirer profit de ces différents types de communication pourrait être la mise en œuvre du protocole Multi-Path TCP (MPTCP). Celui-ci est en effet en cours de standardisation auprès de l'IETF. Multi-Path TCP est une évolution du protocole TCP, autorisant le trafic sur de multiples interfaces et sur différents canaux de communication, de manière simultanée, et ce, pour une même connexion. Le principe de MPTCP est qu'il fonctionne sur la couche transport tout en étant transparent pour les couches inférieures et supérieures. Sa mise en œuvre est facilitée car il est rétrocompatible avec les périphériques standard qui ne supporteraient que le protocole TCP.

Dans les cas de mobilité d'un utilisateur par exemple, MPTCP permet également de faciliter le « handover » qui correspond plus communément au basculement d'un point d'ancrage IP à l'autre. Il offre en plus un intérêt aux problèmes de bande passante mais également dans le cas de la mobilité. En effet, il est possible avec ce protocole de basculer d'un réseau wifi au réseau mobile, et vice versa, lorsque la couverture d'un des moyens de communication devient insuffisante.

En 2013, l'IETF a publié la spécification de Multi Path TCP dans la RFC 6824. [1]

Pour l'instant, celle-ci est toujours au stade expérimental et une mise à jour de la version 1 devrait en principe être approuvée durant le courant de l'année 2019.

1. RFC 6824 https://tools.ietf.org/html/rfc6824

5.2 Avantages de MPTCP

Les différents avantages de MPTCP que nous pouvons citer sont multiples.

En effet, MPTCP offre la possibilité d'autoriser la redondance au niveau de la connexion avec le multiplexage des ressources disponibles. Lorsqu'une des connexions viendrait à ne plus être disponible, il serait toujours possible de basculer sur une connexion restante et disponible sur un autre type de canal. De plus, le multiplexage permet d'améliorer le débit de transmission utile de TCP (throughput) jusqu'à atteindre la somme des débits grâce aux différents canaux physiques.

Dans le cas d'un utilisateur mobile et d'une couverture moindre ou dans le cas d'un déplacement d'un nœud à l'autre, des liens supplémentaires peuvent être ajoutés ou supprimés pour permettre une continuité de la connectivité. Le transfert intercellulaire peut être résolu par l'abstraction de la couche transport sans la nécessité d'ajouter des mécanismes spéciaux dans la couche réseau ou liaison de données.

De plus, il ne nécessite pas de modification au niveau de l'infrastructure pour ces équipements. Les données seront envoyées comme une connexion TCP standard.

Finalement, il est possible d'autoriser l'équilibrage d'une connexion TCP au travers des différentes interfaces disponibles.

5.3 Architecture

MPTCP agit comme une couche intermédiaire entre la partie socket, et une ou plusieurs connexions TCP au sein de la couche transport comme illustré dans la figure 5.1.
MPTCP repose sur un concept de sous-flux que nous aborderons dans la suite de ce chapitre.

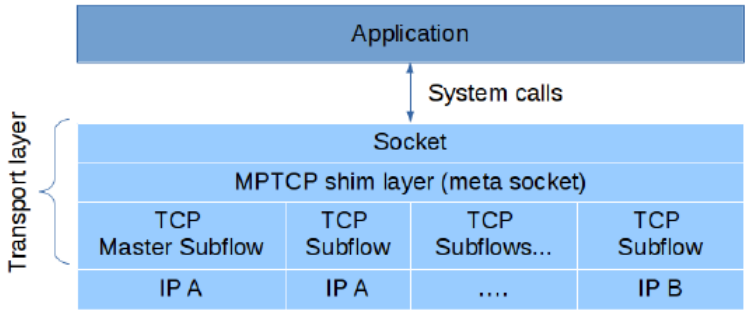


FIGURE 5.1 – Intégration de MPTCP dans la couche transport reproduit à partir de « An implementation of Multipath TCP in ns3 », par COUDRON et SECCI [6]. (avr. 2017) Consulté sur https://hal.sorbonne-universite.fr/hal-01382907

Toutes les informations propres au protocole MPTCP sont stockées dans la partie option de TCP permettant entre autre les opérations suivantes :

- ☞ d'établir une connexion MPTCP via l'opération handshake ;
- ☞ d'ajouter et de retirer des sous flux à une connexion MPTCP ;
- ☞ et finalement de transmettre des données sur la connexion MPTCP.

5.4 Connexion MPTCP et le three-way-handshake

Lors de ce handshake, des clés de 64 bits sont échangées entre l'émetteur et le destinataire de la connexion, et dont le but sera de permettre d'authentifier l'ajout de futurs nouveaux sous flux à cette connexion. A cela, un token d'une longueur de 32 bits est généré en se basant sur une fonction de hashage et sur base de la clé générée précédemment. Cette opération n'est réalisée que lors de l'opération d'initialisation de la connexion.

Toujours lors de l'opération de l'initialisation, une option MP_CAPABLE ainsi que les deux clés sont envoyées par le client lors du troisième message d'initialisation, afin de déterminer si l'hôte prendra en charge le protocole MPTCP. Dans le cas où ce dernier ne le prendrait pas en charge, l'émetteur bascule alors sur l'utilisation standard du protocole TCP en continuant dès lors ces échanges de cette manière. Cela donne donc lieu, dans ce cas, à l'utilisation et l'échange de données via une seule interface pour une même connexion.

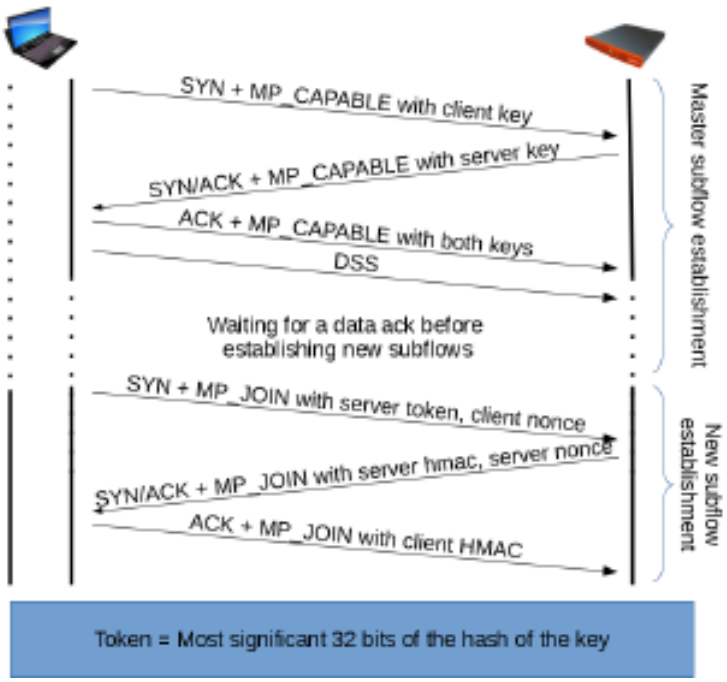


FIGURE 5.2 – Intégration de MPTCP dans la couche transport reproduit à partir de « An implementation of Multipath TCP in ns3 », par COUDRON et SECCI [6]. (avr. 2017) Consulté sur https://hal.sorbonne-universite.fr/hal-01382907

5.4.1 Handshake des sous flux supplémentaires

Une fois l'opération d'initialisation terminée, il est possible dans la suite des échanges de données, d'ajouter ou supprimer de nouveaux sous flux sur l'interface en cours d'utilisation ou sur une toute autre interface. Le client démarre alors à nouveau l'opération d'handshake mais en y intégrant l'option MP_JOIN ainsi que le token généré précédemment. Le nombre de sous flux autorisés n'est pas fixe et peut ainsi fluctuer tout au long de la durée de la connexion.

5.5 Mode de fonctionnement de MPTCP

D'un point de vue fonctionnel, le protocole MPTCP permet d'échanger des données pour une seule connexion via plusieurs interfaces simultanées comme nous l'avons déjà cité précédemment. Cependant, il existe trois types de mode d'opération possibles. Premièrement, il y a ce que l'on appelle le Full-MPTCP. Deuxièmement, le mode backup. Finalement, le single-path.

Mode Full-MPTCP

Le mode de fonctionnement pour le mode Full-MPTCP permet d'utiliser toutes les interfaces de connexion disponibles et opère selon un mécanisme de make-before-break. Cela a comme avantage de maximiser le débit de la connexion mais également d'équilibrer la charge de la connexion.

Mode Backup

Le second disponible est celui du mode Backup. Ce mode dispose également de tous les chemins disponibles pour une connexion cependant, la différence se situe sur le fait que toutes les connexions sont ouvertes. Par contre, un bit indique que le sous flux est de type backup et peut être utilisé dans le cas où aucune autre connexion n'est disponible. Cela permet notamment de réduire les coûts pour n'utiliser que les connexions ayant un coût moindre.

Mode Single-Path

Le dernier des modes de MPTCP est celui du Single-Path ou chemin unique. Dans ce mode, la différence réside dans le fait qu'à tout moment, un seul chemin n'est utilisé pour chaque connexion MPTCP. Lorsqu'un nouveau chemin doit être utilisé pour une raison telle qu'une erreur, une panne ou autre, un nouveau sous flux doit être mis en œuvre, et donc, générant un certain délai au cours de cette opération. De plus, les données transmises au moment où l'erreur se produit sont perdues, ce qui implique que les données doivent à nouveau être envoyées.

Chapitre 6

Continuité de service et de session (SSC)

6.1 SSC et la 5G

Dans les chapitres précédents, nous avons évoqué le concept de « Network Slicing » ou découpage en tranches logiques du réseau 5G. En effet, réunir l'ensemble des exigences destinées à tous les périphériques s'avère quasiment impossible. Si nous prenons le cas des IoT, ceux-ci seront en nombre important mais nécessiteront une bande passante plus faible que pour d'autres types d'usage. En revanche, d'autres nécessiteront par contre une latence plus faible et une bande passante plus élevée.

Afin de pouvoir déployer les réseaux 5G et pouvoir fournir des volumes sans cesse plus important, il est nécessaire de passer entre autres par :
- une gestion de la qualité de service (QoS)
- et une gestion plus efficace du chemin du plan de données utilisateur,

Celles-ci sont essentielles et dépendront des usages. La QoS 5G permet plus de flexibilité par la mise en place d'un ensemble de mécanismes appelés Session and Service Continuity. Un nouveau modèle de QoS est introduit avec la 5G permettant plus de flexibilité et de granularité de cette dernière. Elle autorise des services de données distincts tout en tenant compte les exigences des applications mais aussi par une utilisation plus efficiente des ressources radio.

Pour cela, la QoS s'appuie sur le principe de « Uplink Classifier » (UL-CL) ou classificateur de flux qui a pour fonction de filtrer le trafic de destination selon certaines règles et ainsi orienter le trafic vers certains réseaux locaux, mais également l'optimisation du trafic. L'UE n'a donc pas conscience de ce mécanisme.

La fonction du plan d'utilisateur (UPF) se situant au niveau de la topologie réseau 5G permet de prendre en charge cette fonctionnalité. Son rôle est multiple. Comme nous venons de le voir, l'un des rôles est celui de classer les flux, mais aussi celui de point d'ancrage de session PDU ou de point de branchement PDU.

Dans la pratique, une session PDU établit en fait une connexion entre une application fonctionnant sur l'UE et un réseau de données (Data Network). La continuité de session et de service est en fait associée à un PDU, notion que nous venons d'aborder. Une fois l'association effectuée entre un PDU et un SSC, celui-ci ne changera plus tout au long de la vie du PDU.

Il existe trois types de modes différents de SSC définis pour lesquels il est possible de s'ap-

puyer pour rencontrer les différents types de besoins des applications.

Le premier mode permet de faire en sorte que l'ancrage IP reste fixe et de maintenir un chemin permanent avec l'UE, et cela, malgré une mobilité.

Les modes 2 et 3 autorisent un déplacement de l'ancre IP en définissant deux options différentes :

- ☞ break-before-make (mode SSC 2)
- ☞ make-before-break (mode SSC 3)

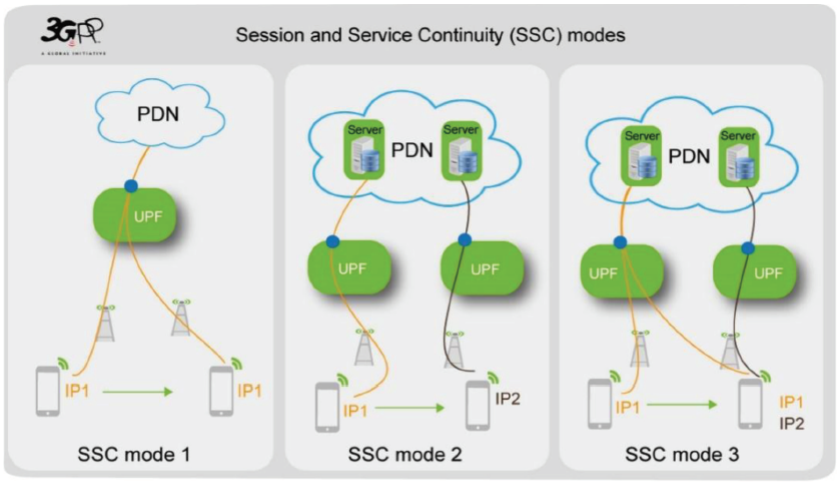


FIGURE 6.1 – Différents modes de continuité des sessions et services reproduit à partir de « The 5G System Architecture », par MADEMANN [26]. (2018) Consulté sur http://www.riverpublishers.com/journal/journal_articles/RP_Journal_2245-800X_615.pdf

6.2 SSC Mode 1

Il s'agit du mode standard avec lequel le point d'ancrage IP reste identique. Il est associé à l'attribution automatique d'une adresse IP fixe. En mode 1, l'adresse IP initiale peut temporairement être complétée par une adresse non persistante. Lors de la mobilité de l'UE, ce changement n'est cependant ni visible pour le protocole MPTCP ni pour une application. Cela autorise le réseau à ajouter ou retirer des point d'ancrage de manière dynamique.

6.3 SSC Mode 2

En mode 2, l'adresse IP initiale est de type non persistante et est remplacée lors de la mobilité avec l'attribution d'une autre IP non persistante. Cela génère un court délai entre les deux basculements. Dans ce mode, le mécanisme de Break-Before-Make (BBM) est exécuté. La continuité de la session peut indiquer à MPTCP qu'une transition BBM est attendue et donc permettre de

suivre et mettre les adresses de backup. Après basculement, un deuxième UPF est créée et le trafic transite alors sur ce nouveau UPF

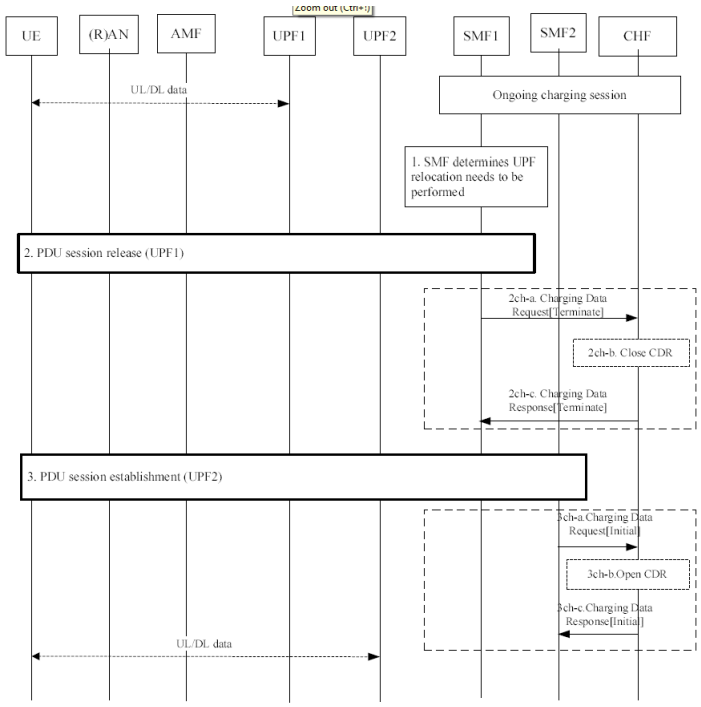


FIGURE 6.2 – Telecommunication management ; Charging management ; Study on charging aspects of 5G system architecture phase 1 reproduit à partir de « 3GPP TR 32.899 V15.0.0 (2018-01) », par PROJECT [33]. (jan. 2018) Consulté sur https://www.3gpp.org/ftp/tsg_SA/WG5_TM/TSGS5_117/SA_79/32899-f10.doc

L'illustration nous présente la séquence du mode 2 lors du basculement d'une UE d'un point d'ancrage IP à l'autre.

6.4 SSC Mode 3

Dans ce dernier mode, la principale différence se situe dans le fait que le mécanisme mis en place est de type Make-Before-Break. Après basculement sur le nouveau point d'ancrage IP, un deuxième UPF est créé. Le trafic transite par l'ancien et le nouvel ancrage IP provisoirement. Un lien vers l'ancien UPF est conversé. L'adresse IP initiale est remplacée par une autre adresse mais il peut y avoir une période de chevauchement. Ce mode offre l'avantage de permettre d'effectuer une transition plus souple lors de de la mobilité.

Deuxième partie

Travail de recherche

Chapitre 7

SSC mode 2 version couche réseau

Dans les chapitres précédents, nous avons introduit le protocole Multipath TCP ainsi que les différents modes de continuité de services et de sessions. Dans la suite de cet exposé, nous nous focaliserons principalement sur le SSC mode 2 . L'objectif étant de reproduire ce mécanisme mais avec un fonctionnement basé sur la couche réseau. Pour rappel, SSC fonctionne quant à lui, sur la couche transport. Une des fonctionnalités clés de SSC est de permettre une continuité de service du terminal que ce soit lors d'un changement d'adresses IP ou d'un changement de point d'ancrage.

7.1 Objectifs

Notre objectif est de transposer en version couche réseau le fonctionnement du mode 2 de SSC. Pour cela, nous allons élaborer notre solution autour de différents concepts. L'objectif premier étant de simuler un UE connecté à un point d'ancrage ayant une zone de service, basculant vers un nouveau point d'ancrage de manière aléatoire et ce, à des moments donnés de manière également aléatoire. L'ensemble des opérations découlant de cette opération seront décrites de manière exhaustive dans la suite de ce chapitre.

7.2 Contexte

Le principe de fonctionnement du mode 2 prévoit que la connectivité puisse être interrompue entre le nœud d'ancrage et l'UE. À ce moment-là, la session PDU établie est libérée et l'adresse IP de l'équipement utilisateur est libérée avant le rattachement au nouveau point d'ancrage. A partir de ce moment, une nouvelle adresse IP différente du premier point d'ancrage est attribuée par ce dernier. Il peut donc y avoir un certain délai entre le début et la fin du basculement.

7.3 Méthodologie de réalisation

Afin de réaliser ces différentes opérations vues précédemment, nous serons amenés à mettre en place différents éléments nécessaires, tel que l'attribution automatique d'adresses IP à notre terminal lors du basculement d'un point d'ancrage à un autre, auquel sera connecté notre UE. Notre projet sera basé sur l'usage du protocole IPv6.

En nous basant sur ce protocole, notre profiterons également des possibilités mises à disposition par IPv6 pour intégrer dans notre travail les extensions d'entête. Celles-ci seront créées par programmation à l'aide du module Scapy. Il s'agit d'un outil permettant d'effectuer des opérations sur notamment des paquets IPv6. Nous l'utiliserons pour créer des paquets permettant de contrôler

la connectivité de l'UE avec l'un des points d'ancrage.

La conception de notre architecture réseau sur laquelle repose notre proof of concept (PoC) sera réalisée à l'aide de l'outil MiniNet, permettant la simulation et l'utilisation d'une architecture réseau de manière virtuelle. À cela, un serveur de type « Router Advertisement » sera également mis en place dans le but de l'attribution automatique d'adresse IPv6.

7.4 Proof of Concept

Notre architecture sera construite à l'aide d'une machine virtuelle fonctionnant avec VMWare Workstation 14 sur laquelle sera installée une distribution Ubuntu 18.04 64 bits. La machine virtuelle disposera d'une configuration composée de 2 vcpu, d'un espace disque de 35Go et 8 Go mémoire.

L'architecture réseau sera construite autour de l'outil MiniNet en version 2.3.0d4. Comme nous l'avons cité précédemment, les adresses IP seront attribuées de manière automatique lors de chaque changement et dépendront du point d'ancrage auquel sera connecté notre nœud. Pour cela, nous utiliserons un serveur Router Advertisement diffusant des préfixes via des annonces RA sur les différentes interfaces auquel ce serveur sera connecté. Ces différents routeurs seront reliés à un switch de type Openflow.

Pour rappel, le protocole Openflow a pour avantage de pouvoir déporter notre plan de contrôle au niveau du contrôleur centralisé et le plan de données au niveau du switch. Le tout fonctionnant sur base d'échange de "packetIn" envoyés vers le contrôleur et des "packet-Out" envoyés par le contrôleur Openflow vers le switch. En pratique, lorsqu'un switch ne possède pas de règle pour le traitement d'un paquet, un paquet In est envoyé au contrôleur qui analyse ses différentes tables. Si une entrée est trouvée dans une des tables, un paquet out est renvoyée vers le switch lui permettant de traiter le paquet. Si aucun règle n'est trouvée, le paquet est droppé ou éliminé tout simplement.

Afin de construire notre PoC, nous l'articulerons autour de différentes parties. La première partie sera chargée des opérations de basculement. La seconde effectuera les tests de connectivité. Finalement, la dernière partie aura pour objet le monitoring du trafic transitant sur les interfaces des différents routeurs.

7.4.1 Scénario de fonctionnement

Le scénario utilisé afin d'illustrer et contrôler le bon fonctionnant du mode 2 SSC sera le suivant : un programme développé en JAVA aura pour fonction d'effectuer la gestion du basculement de l'UE d'un point d'ancrage à l'autre, en traitant toutes les opérations nécessaires au basculement. En parallèle, un script testant la connectivité de l'UE sera utilisé afin de contrôler que la connexion est toujours fonctionnelle. Afin de s'assurer de la connectivité, nous nous baserons sur les commandes de Ping request et reply vers une adresse IPv6.

Au niveau des différents points d'ancrage, des outils de capture de paquets seront utilisés afin de contrôler les informations passant par ces derniers, et par la même occasion, contrôler si le nœud y est connecté et si le trafic y transite bien correctement. Nous pourrons également observer les annonces de routeur en provenance du router advertisement (RS/RA).

Notre scénario sera constitué selon différentes étapes.

- Etape Initialisation
- Etape Fonctionnement normal
- Etape de basculement

7.4.2 Architecture réseau

Au sein de cette architecture, nous avons un terminal qui pourra être relié à l'un des trois routeurs représentés sur le schéma qui suit. Tout le trafic provenant du terminal illustré avec le nom « Hôte B » aura comme passerelle l'adresse IPv6 du routeur auquel il est connecté par son interface. La spécificité de cette architecture créée via l'outil MiniNet, réside dans le fait que le terminal et les routeurs se trouvent dans le même espace de nom que MiniNet. En effet, MiniNet nous permet de créer au sein d'une architecture virtuelle, des équipement et hôtes virtuels ayant chacun sa propre table de routage et interfaces réseaux. Cela est rendu possible grâce à ce que l'on appelle les « network namepaces ». Cela signifie qu'ils ne sont pas visibles depuis la machine virtuelle. Par contre, les interfaces du switch et du contrôleur sont quant à elles dans le même espace de nom que la machine virtuelle et y sont donc visibles. L'intérêt est que l'interface de S1 permet donc de recevoir les annonces routeur diffusées par le service d'annonces routeur (RADVD), qui à son tour les diffuse sur les autres interfaces de S1 en direction de R1, R2, R3 et pour ensuite transiter vers S11, S21 et S31 et arriver finalement sur une des interfaces de B.

Une autre particularité de cette architecture est l'utilisation d'un contrôleur Openflow comme nous l'avons abordé précédemment, que le plan de contrôle des paquets soit déporté sur le contrôleur C0 et le plan de données traité par le switch S1. La version du protocole Openflow utilisé au cours du projet est la version 1.3. En effet, Openflow supporte le traitement, la réécriture des entêtes de paquets IPv6 à partir de cette version et les versions ultérieures.

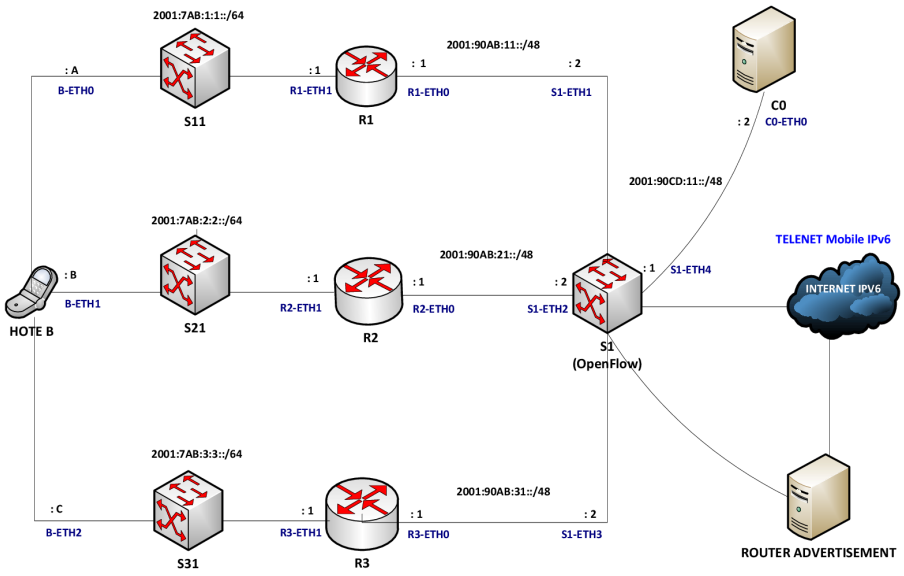


FIGURE 7.1 – Architecture réseau IPv6

7.4.3 Protocole IPv6

Pour rappel, le protocole IPv6 par rapport à son prédécesseur IPv4 possède l'intérêt majeur d'avoir des adresses plus longues codées sur 16 octets permettant de résoudre les problèmes liés à la pénurie d'adresses rencontrées en IPv4.

Le protocole IPv6 autorise un nombre d'adresses Internet de manière pratiquement illimitée. Un exemple trouvé sur le site internet [1] pour illustrer le potentiel en termes d'adresses de l'IPv6 cite l'exemple suivant : « *Précisément, il en a* 2^{128}*, soit approximativement 3 x* 10^{38}*. Si la Terre entière (terre et eau confondues) était couverte d'ordinateurs, IPv6 pourrait allouer 7 x*10^{12} *adresses IP par m². En comparaison, IPv4 permet d'adresser* 2^{32}*=4,29.*10^{9} *adresses.* »

Adressage IP complet de la solution

Nom	Type	Interface	Adresse IP	Passerelle	Remarques
B	Host	B-ETH0 B-ETH1 B-ETH2	2001 : 7AB : 1 : 1 :: /64 2001 : 7AB : 2 : 2 ::/64 2001 : 7AB : 3 : 3 ::/64	default route via 2001 :7ab : 1 : 1 ::1	
R1	Router	R1-ETH0 R1-ETH1	2001 : 90AB : 11 :: 1/48 2001 : 7AB : 1 : 1 :: 1/64	:: / 0 via 2001 : 90ab : 11 ::2 2001 :7ab :1 :1 :: /64 dev r1-eth1	
R2	Router	R2-ETH0 R2-ETH1	2001 : 90AB : 11 :: 1/48 2001 : 7AB : 1 : 1 :: 1/64	:: / 0 via 2001 : 90ab : 21 ::2 2001 :7ab :2 :2 :: /64 dev r2-eth1	
R3	Router	R3-ETH0 R3-ETH1	2001 : 90AB : 31 :: 1/48 2001 : 7AB : 3 : 3 :: 1/64	:: / 0 via 2001 : 90ab : 31 ::2 2001 :7ab :3 :3 :: /64 dev r3-eth1	
S1	Switch	S1-ETH1 S1-ETH2 S1-ETH3	2001 : 90AB : 11 :: 2/48 2001 : 90AB : 11 :: 2/48 2001 : 90AB : 31 :: 2/48 2001 :90cd :11 :: 1 /48	2001 :7ab :1 :1 ::/64 via 2001 :90ab :11 ::1 2001 :7ab :2 :2 ::/64 via 2001 :90ab :21 ::1 2001 :7ab :3 :3 ::/64 via 2001 :90ab :31 ::1 default route via ENS33	Switch Openflow version 1.3
C0	Controller		2001 :90cd :11 :: 2 /48		Contrôleur Openflow 1.3

FIGURE 7.2 – Adressage IPv6

Comparaison structure entête IPv4 et IPv6

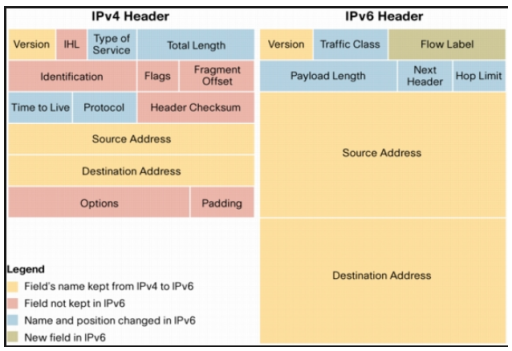


FIGURE 7.3 – Comparaison de structure entre les entêtes IPv4 et IPv6 reproduit à partir de *IPv6 Extension Headers Review and Considerations [IP Version 6 (IPv6)]*, par *IPv6 Extension Headers Review and Considerations [IP Version 6 (IPv6)]* [23]. (oct. 2006) Consulté sur https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

1. https://perso.telecom-paristech.fr/dax/polys/ipv6/ipv6_old.html

Entête et ordre de chaînage

Le test de connexion que nous effectuerons, pour vérifier que la connectivité soit bien maintenue lors des opérations de basculement avec le point d'ancrage et une adresse IPv6 extérieure, sera réalisé via le principe du ping et des paquets ICMPv6. Ce dernier est un protocole formalisé par le RFC 4443 ([2]). Les paquets seront quant à eux construits à l'aide du module Scapy.

Toutes les entêtes utilisées dans ce projet ne sont pas réellement nécessaires mais utilisées à des fins d'expérimentation. Chaque paquet envoyé, sera constitué de la manière suivante :

- d'un paquet IPv6,
- d'une entête HopByHop,
- d'une entête « Destination option »,
- d'une entête de Routage
- d'une entête « Destination option »,
- et finalement un paquet ICMPv6 ou message de diagnostic de type ("echos")

L'ordre dans lequel les paquets IPv6 seront construits sera basé sur l'ordre défini dans la RFC 2460 et illustré ci-après.

L'entête **Hop by Hop** ou « de proche en proche » doit toujours être en première position.

L'entête de **Destination** a la particularité de pouvoir être, par contre, présente 2 fois dans un paquet. L'entête pourra être effectivement positionnée une première fois juste après l'entête « Hop by Hop ». La deuxième sera placée juste après l'extension de chiffrement. Cette entête sera analysée selon sa position dans le paquet IPV6, soit par la destination finale s'il se trouve positionné à la fin, soit pour les routeurs intermédiaires.

L'entête de **Routing** permet d'imposer une route à suivre pour un paquet par rapport à une route traditionnelle. Une liste de serveurs à traverser pourra être spécifiée. A chaque passage au travers d'un de ces routeurs définis dans la liste, le nombre de routeurs à traverser sera décrémenté de 1. De plus, IPv6 autorise de prendre en compte la mobilité d'un nœud via notamment l'utilisation de ce que l'on appelle les extension de routage de type 2 ou Routing Header type 2 (RH2).

En pratique, lorsque le destinataire est un nœud mobile, il est possible que lors de la mobilité, celui-ci passe de son réseau d'origine vers un autre réseau appelé « réseau visité ». Dans ce cas-là, l'adresse de destination contient l'adresse du réseau visité et l'adresse du réseau d'origine est contenue dans cette entête RH2. Lorsque le paquet arrive à destination du mobile, le nœud effectue un remplacement l'adresse contenu dans le champ destination par l'adresse reprise dans l'entête RH2 (Home Adresse (HoA)). Il transmet ensuite le paquet à la couche supérieure.

2. https://tools.ietf.org/html/rfc4443

Order	Header Type	Next Header Code
1	Basic IPv6 Header	-
2	Hop-by-Hop Options	0
3	Destination Options (with Routing Options)	60
4	Routing Header	43
5	Fragment Header	44
6	Authentication Header	51
7	Encapsulation Security Payload Header	50
8	Destination Options	60
9	Mobility Header	135
	No next header	59
Upper Layer	TCP	6
Upper Layer	UDP	17
Upper Layer	ICMPv6	58

FIGURE 7.4 – Recommandation de la RFC 2460 pour l'ordre de chaînage des entêtes IPv6 reproduit à partir de *IPv6 Extension Headers Review and Considerations [IP Version 6 (IPv6)]*, par *IPv6 Extension Headers Review and Considerations [IP Version 6 (IPv6)]* [23]. (oct. 2006) Consulté sur https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

Nous intégrerons cette entête RH2 lors de la construction des paquets IPv6 en y indiquant l'adresse du réseau d'origine et connecté logiquement par l'interface b-eth0 au routeur R1. L'adresse IPv6 du réseau d'origine repris dans cette entête aura le préfixe suivant : « 2001 :7ab :1 :1 : :/64 ».

Dans le cas, où le nœud se trouverait sur un réseau étranger, représenté par une connexion logique et connecté par l'interface b-eth1 ou b-eth2, nous pourrions imaginer que notre solution puisse faire en sorte que l'adresse reprise dans l'entête RH2 soit utilisée et mise dans le champ de destination avant que le paquet ne soit transmis à la couche supérieure. Cela permettrait de rendre l'opération transparente vis à vis des couches supérieures. Celles-ci ne voyant que l'adresse du réseau d'origine. Lorsque le paquet par contre, est émis, c'est l'adresse du réseau visité qui est utilisé comme adresse source, et l'adresse du réseau d'origine placée dans l'entête RH2.

7.4.4 MiNinet - Outils de simulation réseau

Pour construire notre architecture de manière virtuelle et nous permettre d'effectuer notre simulation, nous nous sommes basés sur l'outil MiniNet. La version utilisée est la version 2.3.0d4[38]. Nous avons intégré à notre infrastructure un complément développé par Olivier Tilmans de l'université Catholique de Louvain (UCL)[40] et appelé IPMiniNet.

L'intérêt de ce complément a pour objectif de fournir aux classes qui requièrent de configurer de manière automatique un réseau IPv6 avec les différentes formes de routage.

7.4.5 Scapy - Outils de manipulation de paquets

Pour notre test de connectivité, nous aurions pû nous baser sur l'utilisation du ping standard afin de vérifier le bon fonctionnement de notre solution. Cependant, nous avons décidé de construire le ping à l'aide du module Scapy[3] dans le but de pouvoir manipuler la structure d'un paquet IPv6, nous permettant de la sorte l'ajout d'entêtes supplémentaires tels que les entêtes Hop-ByHop, Destination et Routing.

L'utilisation de l'entête de Routing est essentiel pour permettre de signaler que le nœud et l'adresse d'origine ont changé depuis le dernier basculement.

Structure du paquet IPv6 construit avec Scapy

La figure illustrée ci-après nous présente la structure d'un paquet IPv6 composé des différentes entêtes suivi d'un paquet ICMPv6, utilisé dans notre solution. La destination du paquet ICMPV6 utilisée est l'adresse IP vers les DNS en IPv6 de Google (2001 :4860 :4860 : :8888)

```
###[ IPv6 ]###
  version   = 6
  tc        = 0
  fl        = 0
  plen      = None
  nh        = Hop-by-Hop Option Header
  hlim      = 255
  src       = 2001:7ab:1:1::a
  dst       = 2001:4860:4860::8888
###[ IPv6 Extension Header - Hop-by-Hop Options Header ]###
     nh        = Destination Option Header
     len       = None
     autopad   = On
     \options   \
###[ IPv6 Extension Header - Destination Options Header ]###
        nh        = Routing Header
        len       = None
        autopad   = On
        \options   \
###[ IPv6 Option Header Routing ]###
           nh        = Destination Option Header
           len       = None
           type      = 253
           segleft   = None
           reserved  = 0
           addresses = [  ]
###[ IPv6 Extension Header - Destination Options Header ]###
              nh        = ICMPv6
              len       = None
              autopad   = On
              \options   \
###[ ICMPv6 Echo Request ]###
                 type      = Echo Request
                 code      = 0
                 cksum     = None
                 id        = 0x251
                 seq       = 0x82
                 data      = '    UNamur    2019'
```

FIGURE 7.5 – Paquet IPv6 créé à l'aide de Scapy

7.4.6 Router Advertissement

Notre projet s'appuie une attribution automatique des adresses IP au niveau des interfaces du terminal. Pour cela, nous avons intégré un serveur d'annonces routeur (RADVD). L'intérêt de ce type de ce serveur, est la possibilité de pouvoir configurer une série de paramètres (préfixe réseau, route, ..) qui seront ensuite diffusés sur certaines interfaces, permettant en finale, la configuration automatique. Tout d'abord rappelons qu'en IPv6, les réseaux ont pour simple propriété un préfixe avec par exemple : 2001 :7ab :1 :1 : :/64

Le "/64" signifie que ce sont les 64 bits de poids fort désignent le réseau. 2001 :7ab :1 :1 : :1 désigne la première adresse disponible, et en mettant les 64 bits de poids faible à 1, on obtient donc la dernière adresse, ce qui correspond à 2001 :7ab :1 :1 :ffff :ffff :ffff :ffff

Lorsque l'hote se connecte au réseau, ce dernier émet un paquet type Router Solicitation vers le groupe d'adresses multicast FF02 : : 2 qui est destiné à tous les routeurs.

Le routeur d'annonce émet quant à lui de manière périodique des paquets de type Router Advertisement (RA) à destination du groupe multicast FF02 : : 1 qui correspond par contre à tous les nœuds du lien local. Il émet également un RA en réponse à un paquet RS reçu.

Les paquets de type RS et RA font partie du sous protocole ICMPv6 "Neighbor Discovery" autorisant notamment l'auto-configuration automatique des adresses sans état (SLAAC) en opposition à l'auto-configuration des adresses avec état ou statefull. Dans le cas statefull, les messages RA signalent que le nœud doit obtenir ses informations auprès d'un serveur DHCPv6.

En IPv6, 4 modes d'auto-configuration sont existant :

-	Configuration	Flag M	Flag O
1)	Configuration statique ou nulle	0	0
2)	Stateless Automatic Autoconfiguration (SLAAC) seul	0	0
3)	DHCPv6 (Stateful) avec ou sans SLAAC	1	1
4)	DHCPv6 Stateless avec SLAAC	0	1

Le mécanisme d'auto-configuration d'une interface en IPv6 fonctionne de la manière suivante :

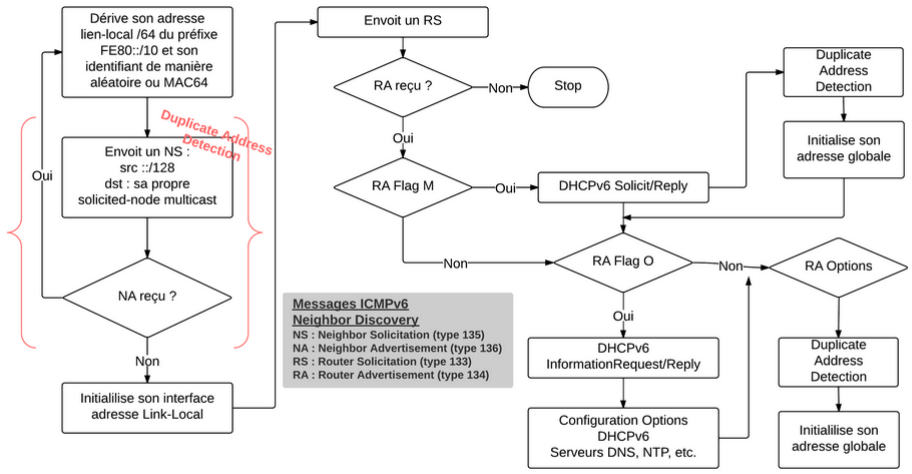
1° Tout d'abord, une adresse de type Lien Local est construite avec le préfixe FE80 : :/10 suivi d'un identifiant d'interface.

2° S'en suit la vérification via le mécanisme DAD (Duplicate Address Detection) qui s'assure que l'adresse n'est déjà pas utilisée. Si aucune réponse n'est reçue, l'adresse devient utilisable. Cela lui permet à l'hôte de communiquer avec le réseau alors qu'en IPv4, cela n'aurait pas été possible car une adresse aurait être demandée auprès d'un serveur DHCP.

3° L'hôte envoie un paquet Router Solicitation à destination des routeurs via le groupe multicast FF02 : : 2.

4° Lors de la réception ou de manière périodique, le routeur d'annonce répond par un Router Advertisement avec des paramètres de configuration RA et Options vers le groupe d'adresses multicast FF02 : : 1.

5° Sur base de ces informations et à l'aide du mécanisme EUI-64, l'adresse est générée.



Mécanisme Stateless Automatic Auto Configuration (SLAAC)

FIGURE 7.6 – Mécanisme d'auto-configuration SLAAC reproduit à partir de *Gestion des adresses IPv6*, par GOFFINET [19]. () Consulté sur https://cisco.goffinet.org/ccna/services-infrastructure/gestion-adresses-ipv6-autoconfiguration-dhcpv6/#42-param%C3%A8tres-router-advertisemen/

7.4.7 Protocole OpenFlow

Comme nous l'avons évoqué précédemment, nous avons fait appel au protocole Openflow pour configurer un contrôleur qui aura pour but de centraliser les règles de routage utilisé par le switch S1. Il permet entre autres d'inspecter les paquets IPv6 et, selon la présence éventuelle d'entêtes spécifiques, de rediriger les paquets vers l'un des ports du switch S1.

Pour rappel, Openflow est un protocole réseau permettant de réaliser une architecture de type Software Defined Network ou réseau défini par logiciel. Nous définirons une série de régles openflow au niveau du contrôleur. Pour cela, nous avons utilisé la version 1.3 d'Openflow autorisant l'inspection des entêtes IPv6 notamment. Le contrôleur utilisé au sein de cette solution reposera sur la plateforme OpenDayLight en version 0.84 (Oxygen).

7.4.8 Connectivité Internet

L'accès Internet est réalisé à l'aide d'une connexion mobile via l'opérateur Telenet. Cet opérateur a été choisi car il nous permet d'avoir une connectivité en IPv6, indispensable pour notre démonstration. Une modification de l'APN a été effectuée afin d'obtenir une connectivité en IPv6, comme nous pouvons l'apercevoir dans le schéma ci-dessous.

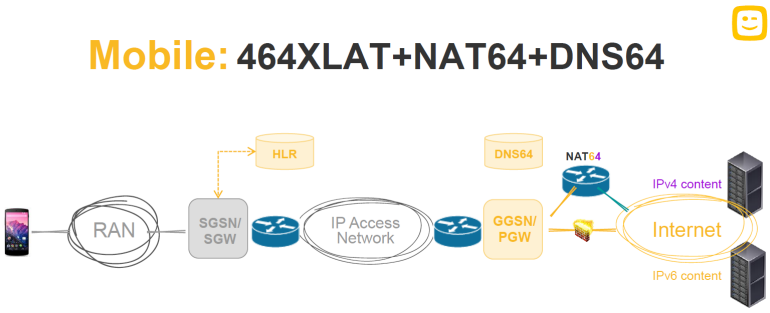


FIGURE 7.7 – Réseau mobile IPv6 Telenet reproduit à partir de *Telenet IPv6 on Mobile 3G/4G*, par THIENPONDT [39]. (mai 2018) Consulté sur https://www.ipv6council.be/IMG/pdf/Telenet_Mobile_IPv6_FUT.pdf

7.5 Exécution et résultats

Au cours de cette section, nous présenterons l'exécution des différentes parties que nous avons pu évoquer jusqu'à présent.

L'exécution de la solution s'articule selon différentes étapes :
- une étape d'initialisation,
- une étape en fonctionnement normal, c'est-à-dire sans opération particulière ou de basculement
- et finalement l'étape de basculement entre deux points d'ancrage du terminal.

L'ensemble du développement s'est déroulé autour d'un développement en JAVA et en PYTHON. Le choix du développement du module de gestion du basculement a été réalisé sur base de langage Java, pour les capacités qu'offre ce langage. La finalité de ce programme est la reproduction du fonctionnement du mode 2 SSC.

Les parties relatives à la création de l'architecture réseau, du ping à l'aide du module Scapy, ont quant à elles été réalisées en nous basant sur le langage PYTHON.

Développement JAVA

Notre programme JAVA comporte une classe nommée NetInfo, destinée à stocker les informations relatives à chaque interface du terminal telles que les adresses IP, le nom de chaque interface, la mac adresse (MAC) et d'autres informations utiles au fonctionnement du programme.

Lors du démarrage du programme, le programme initialise une série de paramètres tels que l'interface b-eth0 comme interface par défaut, la route par défaut et désactive les interfaces b-eth1 b-eth2.

Remarque, il aurait possible de n'avoir qu'une seule interface définie au sein du terminal. Cependant, afin de faciliter la réalisation, le choix a été fait de définir deux interfaces supplémentaires, chacune pouvant être connectée à un des autres routeurs accessibles. En effet, nous avons construit notre architecture réseau avec 3 routeurs et de façon à ce que le nœud puisse se connecter à un seul routeur à la fois.

Dans le but de communiquer avec les autres applications développées en PYTHON, nous avons opté pour l'utilisation d'un fichier d'échange qui sera mis à jour par le programme JAVA

et lu par les scripts en PYTHON. Les informations utilisées au sein de ce fichier comprennent le nom de l'interface actuellement connectée, l'adresse IP de celle-ci et le nom et l'adresse IP de l'interface précédente. Ces informations seront notamment utiles pour signaler aux nœuds du réseau que la connexion a changé et seront utilisées lors du retour d'un paquet par le switch S1 et pour son traitement en réécrivant l'adresse de destination lors de la présence d'entête spécifique.

Afin d'automatiser le basculement entre deux points d'ancrage, l'opération est effectuée à des moments déterminés, et ce, de manière aléatoire. Le choix du nouveau point d'ancrage du terminal est également déterminé de manière aléatoire.

Les annonces RA sont également utilisées pour configurer l'interface lors des opérations de basculement.

7.5.1 Exécution

Pour poursuivre notre chapitre, nous présenterons les différentes étapes d'exécution de notre programme ainsi que les différents points importants à expliciter.

L'objectif de la présentation de l'exécution de la solution est de montrer que le terminal maintient sa connexion malgré la bascule de son point d'ancrage.

Lors du lancement de la solution, nous initialisons notre interface b-eth0 comme interface de départ. Nous faisons en sorte que cette interface ne dispose pas d'adresse et que celle-ci soit configurée à l'aide des paquets RA comme l'illustre l'image qui suit.

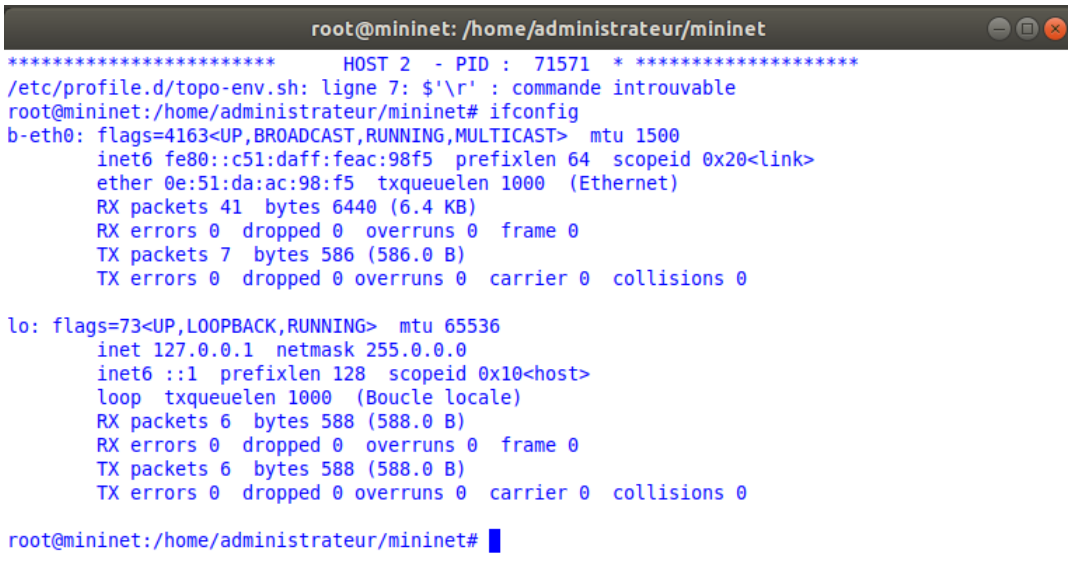


FIGURE 7.8 – Configuration initiale de l'interface b-eth0

Lors du démarrage, s'en suit l'envoi de paquets Router Solicitation depuis l'interface b-eth0 destiné au groupe multicast FF02 : :2, c'est-à-dire à destination de tous les routeurs.

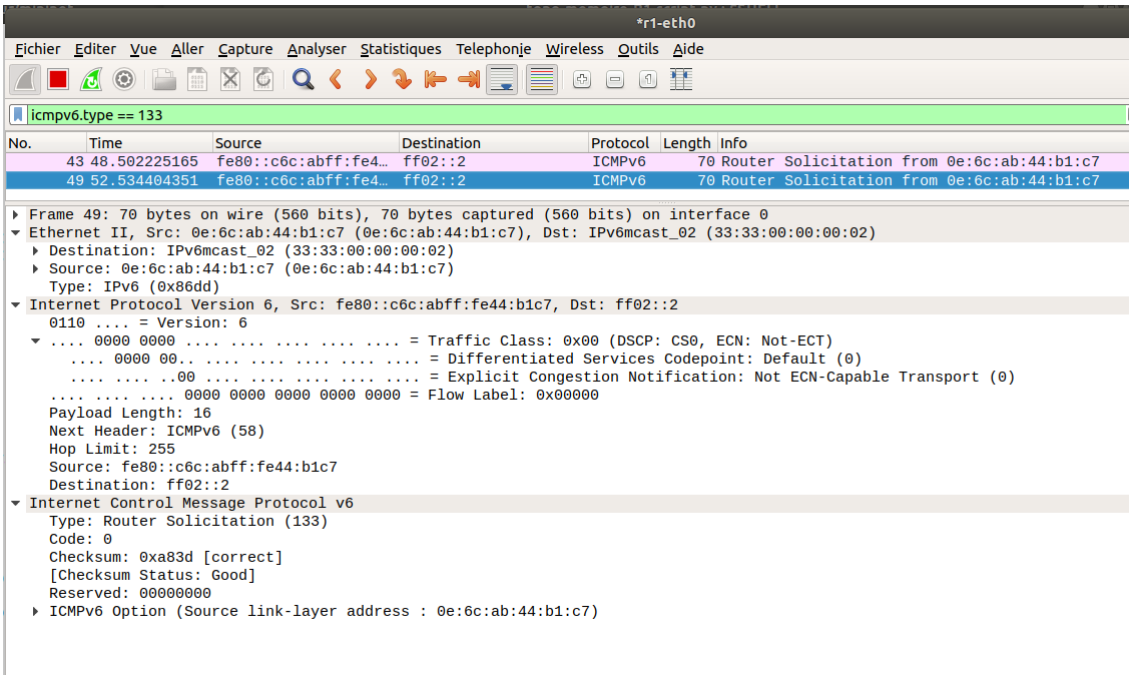


FIGURE 7.9 – Contenu d'un paquet Router Solicitation envoyé depuis b-eth0

Soit de manière périodique ou en réponse à ces paquets, un paquet Router Advertisement est envoyé au groupe multicast FF02 : :1, qui correspond par contre à tous les hôtes du lien réseau.

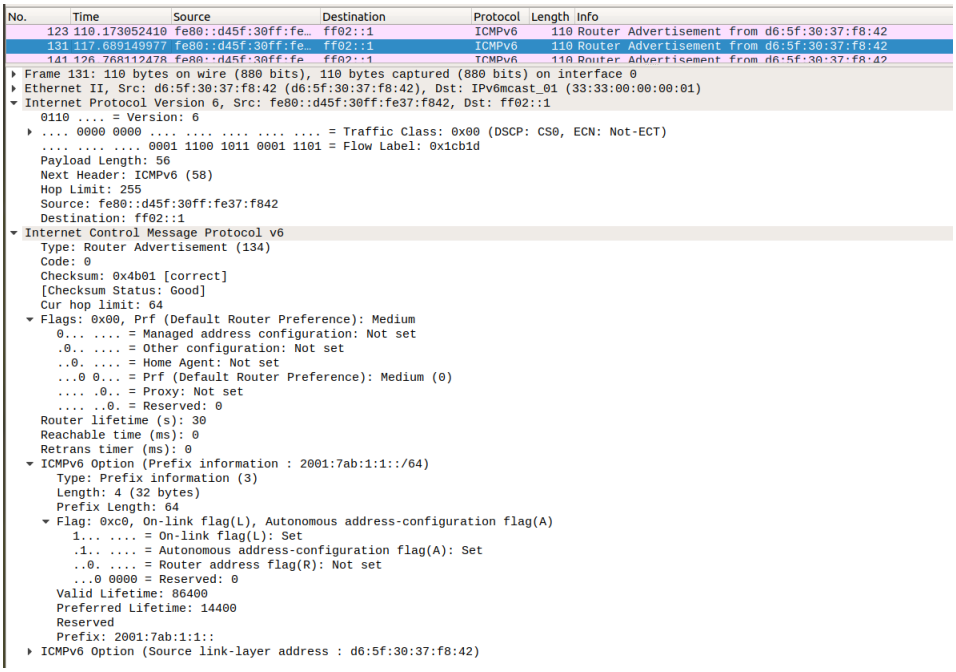


FIGURE 7.10 – Contenu d'un Router Advertisement reçu par l'interface b-eth0 et envoyé depuis le serveur RADVD

Lors de la réception en retour d'un paquet Router Advertisement, celui-ci contient une série d'informations qui seront nécessaires à la configuration de l'interface. Parmi ceux-ci, nous pouvons citer notamment la présence du préfixe réseau, la route par défaut. Les RA peuvent également

informer l'hôte d'utiliser un serveur DHCPv6.

Le préfixe réseau est utilisé par l'interface b-eth0 pour s'auto-configurer. Avec les options ICMPv6 reprises dans l'illustration ci-dessus, nous pouvons constater dans notre cas que seuls les flags correspondant à la configuration automatique (Autonomous-address-configuration) et On-Link sont positionnés à 1. Ce qui correspond bien à ce que nous avons défini dans notre configuration reprise dans le fichier radvd.conf et utilisé par notre serveur d'annonces routeur lors de la diffusion des RA.

```
interface r2-eth1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 2001:7ab:2:2::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr off;
        };
};
```

FIGURE 7.11 – Configuration diffusée par le serveur RADVD

Le préfixe représente une partie de l'information nécessaire à la configuration de notre interface. Un deuxième mécanisme nommé "Extended Unique Identifier" ou "identifiant unique étendu" (EUI-64) entre en jeu.

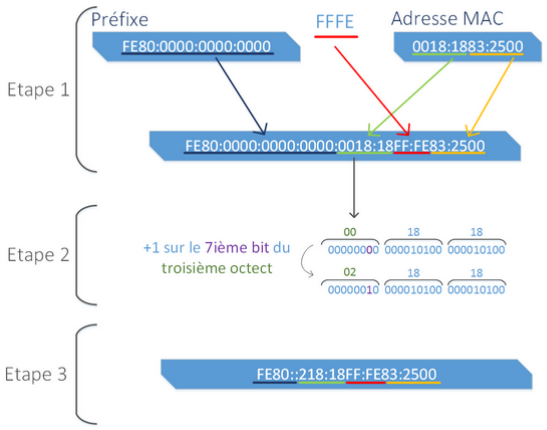


FIGURE 7.12 – Calcul d'une adresse IPv6 en EUI-64 reproduit à partir de *IPv6 : Qu'est ce que l'EUI-64*, par DORIGNY [10]. (fév. 2015) Consulté sur https://www.it-connect.fr/ipv6-quest-ce-que-leui-64-13

Lors de la réception du préfixe réseau, trois opérations sont effectuées lors de la combinaison du préfixe et du mécanisme EUI-64.

Tout d'abord, on utilise le préfixe "FE80 :0000 :0000 :0000" ainsi que la mac adresse de l'interface que l'on combine ensemble.

On combine donc le préfixe et les trois premiers octets de l'adresse MAC (MAC) suivant le FFFE ainsi que les trois derniers octets de la MAC.

Une modification a lieu sur le 7ème bit du 3ème octet en effectuant une opération de "+1" sur la valeur décimale. Finalement, l'adresse IP est obtenue en supprimant les "0" inutiles.

Au final, nous pouvons constater que l'interface a bien été configurée avec le préfixe réseau ainsi que sur base du mécanisme EUI-64

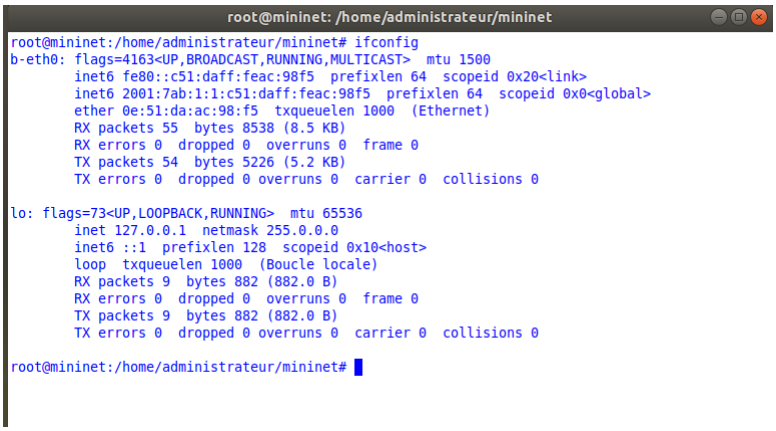


FIGURE 7.13 – Interface b-eth0 avec l'adresse IPv6 configurée

Etape de fonctionnement normal

Après l'étape d'initialisation de la solution, celle-ci entre dans ce que l'on peut appeler l'étape de fonctionnement normal. Ils s'agit à ce moment-là des échanges de données entre le terminal et le point d'ancrage sans opération de basculement. Durant cette étape, les données transitent par le nœud auquel est connecté notre terminal.

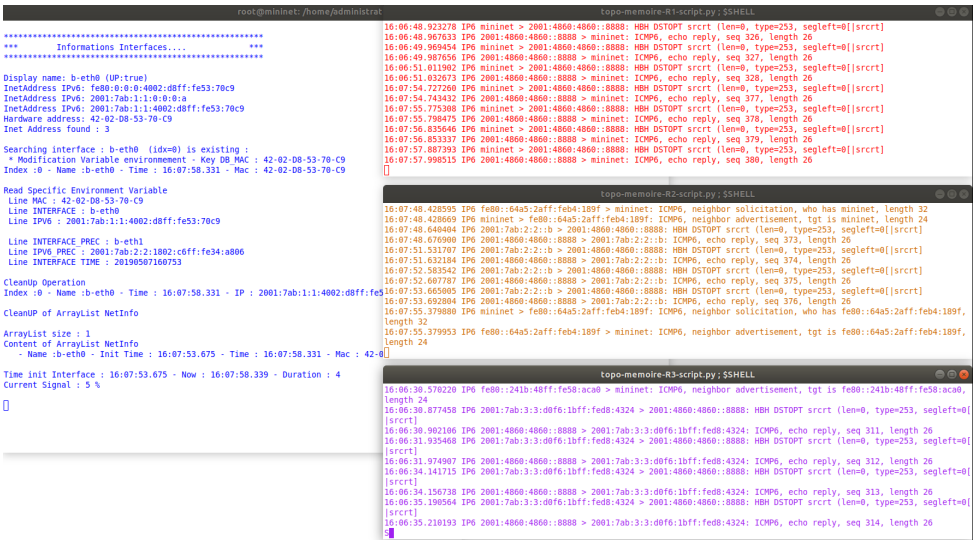


FIGURE 7.14 – Basculement du trafic du point d'ancrage 2 à 1

Pour simuler un échange de trafic depuis notre terminal et notre point d'ancrage, nous utilisons la commande Ping permettant de tester la connectivité avec une autre machine. Pour ce faire, nous avons conçu une version du Ping à l'aide du module Scapy. Notre script construit un paquet ICMPv6 constitué d'une série d'entêtes IPv6 comme nous pouvons l'examiner sur l'illustration ci-dessous. L'adresse IP de destination utilisée est celle correspondant aux DNS IPv6 de Google

(2001 :4860 :4860 : :8888).

```
###[ IPv6 ]###
  version   = 6
  tc        = 0
  fl        = 0
  plen      = None
  nh        = Hop-by-Hop Option Header
  hlim      = 255
  src       = 2001:7ab:1:1::a
  dst       = 2001:4860:4860::8888
###[ IPv6 Extension Header - Hop-by-Hop Options Header ]###
     nh        = Destination Option Header
     len       = None
     autopad   = On
     \options   \
###[ IPv6 Extension Header - Destination Options Header ]###
        nh        = Routing Header
        len       = None
        autopad   = On
        \options   \
###[ IPv6 Option Header Routing ]###
           nh        = Destination Option Header
           len       = None
           type      = 253
           segleft   = None
           reserved  = 0
           addresses = [  ]
###[ IPv6 Extension Header - Destination Options Header ]###
              nh        = ICMPv6
              len       = None
              autopad   = On
              \options   \
###[ ICMPv6 Echo Request ]###
                 type      = Echo Request
                 code      = 0
                 cksum     = None
                 id        = 0x251
                 seq       = 0x82
                 data      = '    UNamur    2019'
```

FIGURE 7.15 – Paquet IPv6 créé à l'aide de Scapy

Afin de nous assurer que notre trafic transite bien hors de la machine virtuelle, nous lançons l'outil Wireshark pour capturer les paquets sur notre interface Wifi connectée à Internet. La capture d'écran nous illustre le trafic vers la destination 2001 :4860 :4860 : :8888, ce qui correspond effectivement au but recherché.

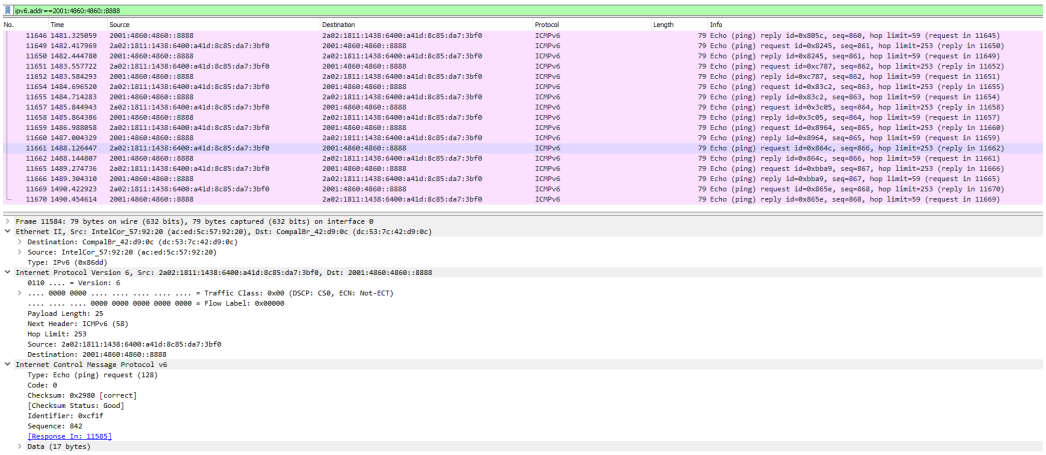


FIGURE 7.16 – [État du trafic transitant par l'interface physique et vers Internet

Étape de basculement

Afin de rendre notre basculement automatique et de manière aléatoire, nous avons ajouté à notre solution une valeur représentant le niveau du signal de connexion avec le point d'ancrage. Celui-ci fluctue de manière continue et lorsque celui-ci descend en dessous de 20 %, cela provoque un basculement vers un nouveau point d'ancrage. Le choix du nouveau point d'ancrage est également choisi de manière aléatoire, sur base des nœuds disponibles.

Lorsque l'étape de basculement a lieu, une série d'opérations sont effectuées par le programme sur l'interface du terminal en question.

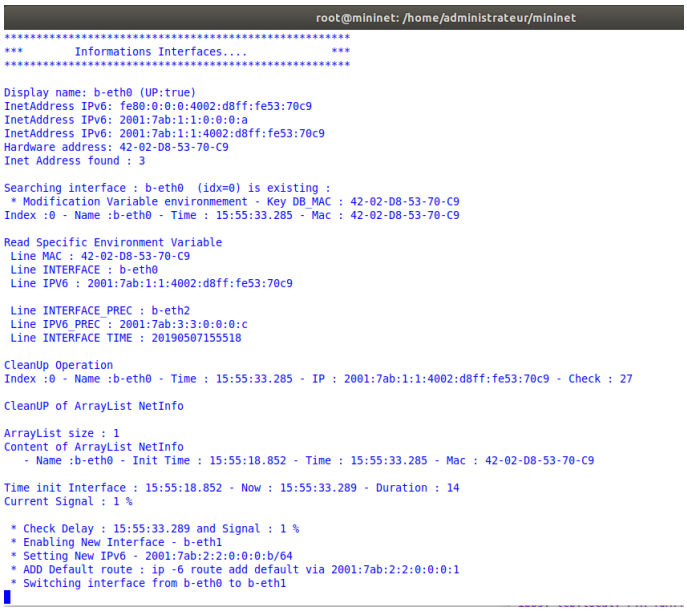


FIGURE 7.17 – Basculement du terminal d'un point d'ancrage à l'autre

Tout d'abord, lors de cette opération, la première séquence effectuée est la suppression de la route par défaut configurée sur le terminal afin d'éviter que les paquets continuent de transiter par le point d'ancrage actuel. Ensuite l'interface est connectée vers le nouvel point d'ancrage. Suite à cela, a lieu la configuration automatique de l'adresse IP à l'aide des échange RS/RA.

Finalement, une nouvelle route par défaut est configurée afin de rediriger les paquets vers le nouveau point d'ancrage.

Lors de cette opération, l'écran principal nous affiche les opérations qui ont lieu lors de ce basculement.

Dans les images qui suivent, on peut visualiser les captures d'écran des paquets transitant par le routeur 3. On peut constater avec, les numéros de séquences des paquets, de la continuité de ceux-ci lors du passage sur l'autre point d'ancrage. Le dernier échange sur le point d'ancrage précédent se termine à la séquence 24 et à l'heure de 17 :00 :49 :277545. Le nouvel échange reprend avec la séquence n° 25 et à l'heure 17 :00 :50 :312317, ce qui nous donne un délai de basculement de l'ordre d'une seconde.

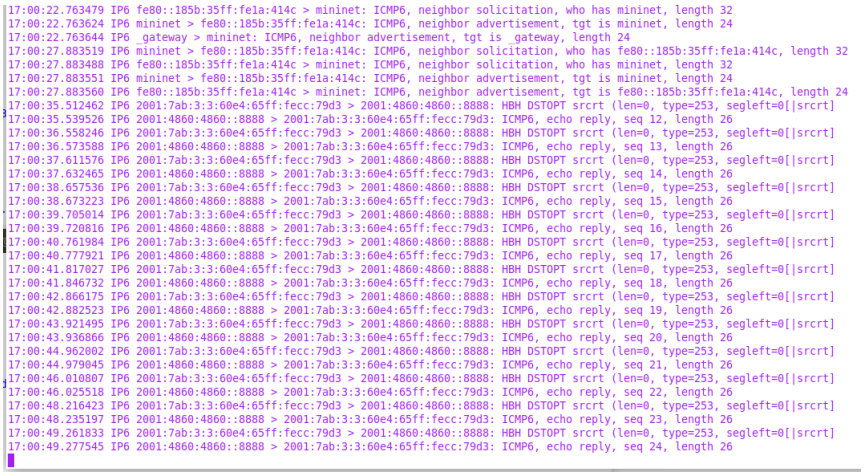


FIGURE 7.18 – Capture d'écran avant le basculement du trafic du point d'ancrage 3 à 2

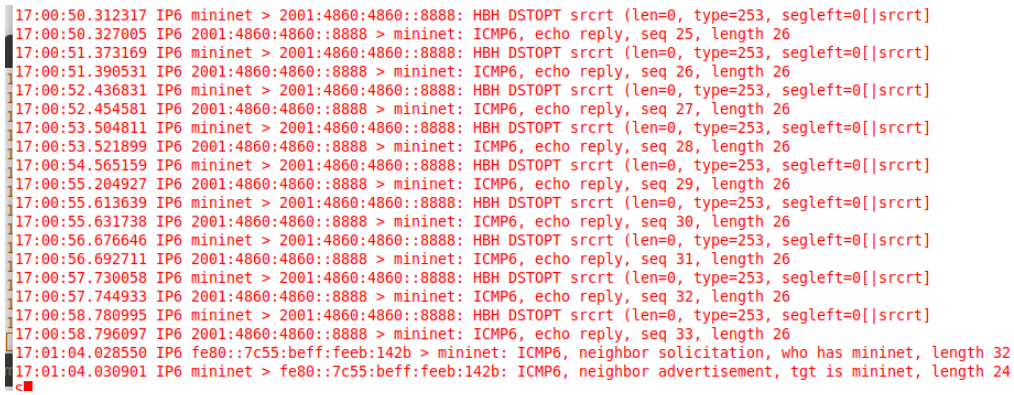


FIGURE 7.19 – Capture d'écran après le basculement du trafic du point d'ancrage 3 à 2

Chapitre 8

Conclusion

Tout au long de la réalisation de ce travail, nous avons proposé une analyse des éléments qui seront mis en place dans le cadre de la future norme mobile 5G. Nous avons également prolongé ce chapitre consacré à la 5G en abordant le protocole MPTCP, les mécanismes de Session and Service Continuity et de manière plus détaillé le mode 2 SSC.

8.1 Enseignement

C'est dans ce contexte de mode 2 SSC, que nous avons proposé la réalisation d'une version fonctionnant en mode couche réseau. Ce développement nous a permis d'illustrer d'une part, le fonctionnement de SSC mode 2, et d'autre part, en y intégrant un serveur d'annonces routeur, l'utilisation du protocole IPv6, l'utilisation de Mininet pour la création de l'architecture réseau et de Scapy pour la anipulation de paquets notamment IPv6.

Pour finir, nous avons également associé au développement de ce projet, l'utilisation du protocole Openflow pour une gestion du plan de contrôle et de données des switches.

8.2 Problèmes rencontrés (problèmes des ra, freeze, . . .)

8.2.1 RA

Lors de la mise en place de notre serveur RA, nous avons rencontré des difficultés de par notre architecture, avec la diffusion des RA jusqu'à l'interface de l'hôte final. Cela s'est traduit par un problème de réception des RA.

8.2.2 Transit

A cela, durant les opérations de basculement du terminal d'un routeur à l'autre, une autre difficulté a été de continuer à faire transiter les données d'une interface à l'autre. S'est alors posé le problème de la communication entre l'application développée en Java et les scripts écrits en PYTHON. La solution a été de faire communiquer les applications à l'aide d'un fichier d'échange. A la base, un mécanisme basé sur l'utilisation des variables d'environnement a été testé mais cela ne s'est pas avéré concluant. En effet, l'utilisation des variables au niveau du système d'exploitation Linux et celles utilisées lors de l'exécution des Python et JAVA n'a pas permis d'arriver à un résultat satisfaisant.

Finalement, la solution s'est portée sur l'écriture d'un fichier d'échange utilisé par le programme Java pour indiquer les interfaces actuelles et précédentes notamment, et la lecture de ces

informations par le script PYTHON pour la création des paquets ICMPv6.

8.2.3 Freeze

D'autres problèmes ont été constatés comme des cas freezes lors du basculement. Ceux-ci étaient dus à l'utilisation simultanée du ficher d'échange en écriture par le programme Java et la lecture par le script. La solution a été d'effectuer une copie du fichier et de réserver la copie pour la lecture des données.

En utilisant la création des paquets IPv6 avec le module Scapy, nous avons également constaté un dysfonctionnement avec le ping. En effet, il était possible d'effectuer une demande d'écho mais sans obtenir la réponse en retour. Le problème se situait dans la conception du paquet et dans les entêtes IPv6 utilisés.

8.3 Future works

Actuellement, au moment de l'écriture de ce document, réside un dernier problème avec l'utilisation du protocole Openflow entre le contrôleur et le switch s1. L'objectif étant la finalisation de cet aspect avant la démonstration.

8.3.1 Aspects sécuritaires

Un des aspects n'ayant pas été pris en compte durant la réalisation de notre développement est celui de l'aspect sécuritaire. L'ajout de cet aspect pourrait faire l'objet d'un « future works » par l'utilisation et l'échange de tokens entre le terminal et le point d'ancrage afin de s'assurer de l'authenticité de l'émetteur et du récepteur.

De plus, afin de rendre les échanges également plus sécurisés, il serait également envisageable d'utiliser des protocoles de type IPsec entre les points d'ancrage et le terminal par exemple.

Bibliographie

[1] 5G-ACIA. *5G for Connected Industries and Automation*. Nov. 2018. URL : https://www.5g-acia.org/fileadmin/5G-ACIA/Publikationen/Whitepaper_5G_for_Connected_Industries_and_Automation/5G-for-Connected-Industries-and-Automation-White-Paper.pdf.

[2] DNS BELGIUM. *Clair et court : tout ce que vous devez savoir sur IPv6*. French. Rapp. tech. DNS Belgium, 2018, p. 1. URL : https://www.dnsbelgium.be/fr/nouvelles/clair-et-court-tout-ce-que-vous-devez-savoir-sur-ipv6 (visité le 17/04/2019).

[3] Philippe BIONDI et the SCAPY COMMUNITY. *Scapy - Packet crafting for Python2 and Python3*. Anglais. Rapp. tech. Philippe Biondi et the Scapy community. URL : https://scapy.net/index.

[4] Kalyani BOGINENI, Arashmid AKHAVAIN, Tom HERBERT, Dino FARINACCI, Alberto RODRIGUEZ-NATAL, Giovanna CAROFIGLIO, Jordan AUGE, Luca MUSCARIELLO, Pablo Camarillo GARVIA et Shunsuke HOMMA. *Optimized Mobile User Plane Solutions for 5G*. Internet-Draft draft-bogineni-dmm-optimized-mobile-user-plane-01. Work in Progress. Internet Engineering Task Force, juin 2018. 65 p. URL : https://datatracker.ietf.org/doc/html/draft-bogineni-dmm-optimized-mobile-user-plane-01.

[5] GAILLOT Davy CHALLITA FRÉDÉRIC LIENARD Martine. *Les réseaux massifs : une solution pour la 5G ?* French. Sous la dir. de TELICE-IEMN (Cité SCIENTIFIQUE). 2016. URL : www.cnfm.fr/VersionFrancaise/animations/JNRDM/JNRDM'2017_Papiers/Energie&IoT/1_13_Fr%C3%A9d%C3%A9ric%20CHALLITA.pdf.

[6] Matthieu COUDRON et Stefano SECCI. « An implementation of Multipath TCP in ns3 ». In : *Computer Networks* 116 (avr. 2017), p. 1-11. DOI : 10.1016/j.comnet.2017.02.002. URL : https://hal.sorbonne-universite.fr/hal-01382907.

[7] Sabine DAHMEN-LHUISSIER. *Mobile, LTE, 5G, building blocks, MEC, multi-access Edge Computing, NFV Network Functions Virtualisation, NGP, mWT*. Nov. 2018. URL : https://www.etsi.org/technologies/5g.

[8] Berna Özbek DIDIER LE RUYET. *Systèmes MIMO et codage spatio-temporel*. French. Rapp. tech. CNAM, 2005, p. 8. URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.1720&rep=rep1&type=pdf (visité le 15/07/2018).

[9] Nikhil Vyakaranam DILIP KRISHNA S. *5G, Architecture, Cloud, Internet Of Things, Network Slicing*. English. URL : https://cloudifynetwork.com/5g-network-slicing/.

[10] Mickael DORIGNY. *IPv6 : Qu'est ce que l'EUI-64*. Fév. 2015. URL : https://www.it-connect.fr/ipv6-quest-ce-que-leui-64-13.

[11] Jérôme DURAND. « Segment Routing L'art de faire plus avec moins ». In : *JRES - Nantes* (2018).

[12] EFORT. « Le Réseau Coeur 5G ». In : (). Sous la dir. d'EFORT. URL : http://www.efort.com/r_tutoriels/RESEAU_COEUR_5G_EFORT.pdf (visité le 28/06/2018).

[13] Alan FORD, Costin RAICIU, Mark HANDLEY et Olivier BONAVENTURE. *TCP Extensions for Multipath Operation with Multiple Addresses*. Anglais. RFC 6824. 2070-1721. IETF, jan. 2013. DOI : 10.17487/RFC6824. URL : https://tools.ietf.org/rfc/rfc6824.txt.

[14] The Linux FOUNDATION. *Wiki OpenDaylight*. Anglais. Rapp. tech. The Linux Foundation. URL : https://wiki.opendaylight.org.

[15] Association G6. *La gestion de la mobilité IPv6*. Mar. 2006. URL : http://livre.g6.asso.fr/index.php/La_gestion_de_la_mobilit%C3%A9_IPv6.

[16] Association G6. *Traitement optimisé des extensions*. Rapp. tech. Association G6, juin 2015. URL : http://livre.g6.asso.fr/index.php/IPv6_QO.

[17] Galileo GALILEI. *Inspirational Quotations by Galileo Galilei (Italian Astronomer)*. URL : http://www.inspiration.rightattitudes.com/authors/galileo-galilei/.

[18] C. A. GARCIA-PEREZ et P. MERINO. « Enabling Low Latency Services on LTE Networks ». In : *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. Sept. 2016, p. 248-255. DOI : 10.1109/FAS-W.2016.59. URL : https://www.researchgate.net/profile/Cesar_Garcia_Perez/publication/311716854_Enabling_Low_Latency_Services_presentation/links/5857ac3d08ae8f695559f337/Enabling-Low-Latency-Services-presentation.pdf?origin=publication_list.

[19] François GOFFINET. *Gestion des adresses IPv6*. Rapp. tech. URL : https://cisco.goffinet.org/ccna/services-infrastructure/gestion-adresses-ipv6-autoconfiguration-dhcpv6/#42-param%C3%A8tres-router-advertisemen/.

[20] Tom HERBERT et Kalyani BOGINENI. *Identifier Locator Addressing for Mobile User-Plane*. Internet-Draft draft-herbert-ila-mobile-01. Work in Progress. Internet Engineering Task Force, mar. 2018. 22 p. URL : https://datatracker.ietf.org/doc/html/draft-herbert-ila-mobile-01.

[21] HUAWEI. *5G Network Architecture A High-Level Perspective*. English. Sous la dir. d'HUAWEI. URL : https://www.huawei.com/en/industry-insights/outlook/mbb-2020/trends-insights/5g-network-architecture.

[22] HUAWEI. *White Paper: 5G Network Architecture - A High-Level Perspective - Industry insight in Huawei*. Déc. 2016. URL : https://www.huawei.com/en/industry-insights/outlook/mobile-broadband/insights-reports/5g-network-architecture.

[23] *IPv6 Extension Headers Review and Considerations [IP Version 6 (IPv6)]*. Oct. 2006. URL : https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html.

[24] « Le paquet Router Advertisement ». In : (). URL : http://blog.bashy.eu/2011/03/paquet-router-advertisement/ (visité le 14/04/2019).

[25] *Les nouveaux enjeux de la 5G*. French. Rapp. tech. ARCEP, mar. 2017, p. 43. URL : https://www.arcep.fr/uploads/tx_gspublication/rapport-enjeux-5G_mars2017.pdf (visité le 01/07/2018).

[26] Frank MADEMANN. « The 5G System Architecture ». In : *Journal of ICT Standardization* 6.1 (2018), p. 77-86. URL : http://www.riverpublishers.com/journal/journal_articles/RP_Journal_2245-800X_615.pdf.

[27] Alain Sibille MARCEAU COUPECHOUX. *La 5G: défis, fondamentaux et innovations*. French. Sous la dir. de Télécom PARISTECH. URL : http://innovation-regulation2.telecom-paristech.fr/wp-content/uploads/2017/07/1-Marceau-Coupechoux-Chaire_IRSN_26-09-2017.pptx.pdf.

[28] Patrick MARSCH, Icaro Leonardo Da SILVA, Ömer BULAKCI, Milos TESANOVIC, Salah Eddine El AYOUBI, Thomas ROSOWSKI, Alexandros KALOXYLOS et Mauro BOLDI. « 5G Radio Access Network Architecture: Design Guidelines and Key Considerations ». In : *IEEE Communications Magazine* 54.11 (2016), p. 24-32. DOI : 10.1109/MCOM.2016.1600147CM. URL : https://doi.org/10.1109/MCOM.2016.1600147CM.

[29] Claire MASTERSON. *Massive MIMO and Beamforming: The Signal Processing Behind the 5G Buzzwords*. English. Rapp. tech. Analog Dialogue 51-06, June 2017, juin 2017, p. 5. URL : http://www.analog.com/media/en/analog-dialogue/volume-51/number-3/articles/massive-mimo-and-beamforming-the-signal-processing-behind-the-5g-buzzwords.pdf (visité le 01/07/2018).

[30] Christoph PAASCH, Gregory DETAL, Fabien DUCHENE, Costin RAICIU et Olivier BONAVENTURE. « Exploring Mobile/WiFi Handover with Multipath TCP ». In : *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*. CellNet '12. Helsinki, Finland : ACM, 2012, p. 31-36. ISBN : 978-1-4503-1475-6. DOI : 10.1145/2342468.2342476. URL : http://doi.acm.org/10.1145/2342468.2342476.

[31] Imtiaz PARVEZ, Ali RAHMATI, Ismail GÜVENÇ, Arif I. SARWAT et Huaiyu DAI. « A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions ». In : *CoRR* abs/1708.02562 (2017). arXiv : 1708.02562. URL : http://arxiv.org/abs/1708.02562.

[32] Autorité de régulation des communications électroniques et des POSTES. *Les nouveaux enjeux de la 5G*. French. Rapp. tech. ARCEP, mar. 2017, p. 43. URL : https://www.arcep.fr/uploads/tx_gspublication/rapport-enjeux-5G_mars2017.pdf (visité le 01/07/2018).

[33] 3rd Generation Partnership PROJECT. « 3GPP TR 32.899 V15.0.0 (2018-01) ». In : *3GP - Telecommunication management; Charging management;Study on charging aspects of 5G system architecture phase 1* (jan. 2018). 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Charging management; Study on charging aspects of 5G system architecture phase 1 (Release 15). URL : https://www.3gpp.org/ftp/tsg_SA/WG5_TM/TSGS5_117/SA_79/32899-f10.doc.

[34] Karim RABIE. *Core Network Evolution - 5G Service based Arhcitecture*. Déc. 2017. URL : https://netmanias.com/en/post/blog/12967/5g/core-network-evolution-5g-service-based-arhcitecture.

[35] ROHDE et SCHWARZ. *Formes d'ondes envisageables pour la 5G*. French. Sous la dir. de ROHDE et SCHWARZ. URL : https://www.rohde-schwarz.com/ch-fr/de-detection-superieures/test-and-measurement/wireless-communication/5g/formes-ondes-5g/formes-ondes%5C-5g%5C_230224.html.

[36] Sidhartha Sankar SAHOO, Malaya Kumar HOTA et Kalyan Kumar BARIK. *5G Network a New Look into the Future: Beyond all Generation Networks*. Jan. 2014. URL : http://pubs.sciepub.com/ajss/2/4/5/index.html (visité le 17/12/2018).

[37] Dr. Harrison J. SON et Chris YOO. *E2E Network Slicing - Key 5G technology : What is it? Why do we need it? How do we implement it?* Nov. 2015. URL : https://www.netmanias.com/en/?m=view&id=blog&no=8325.

[38] Mininet TEAM. *Mininet - An Instant Virtual Network on your Laptop*. Anglais. Rapp. tech. Mininet Team. URL : http://mininet.org/.

[39] Hans THIENPONDT. *Telenet IPv6 on Mobile 3G/4G*. Mai 2018. URL : https://www.ipv6council.be/IMG/pdf/Telenet_Mobile_IPv6_FUT.pdf.

[40] Olivier TILMANS. *Experimenting with Mininet and IPv6 routes*. Anglais. Rapp. tech. UCLouvain, nov. 2017. URL : https://obonaventure.github.io/cnp3blog/ipmininet/.

[41] Anne WEI. *4G LTE (Long Term Evolution)*. French. Sous la dir. de CNAM PARIS. 2011. URL : http://www.academia.edu/5964059/4G_LTE_Long_Term_Evolution.

[42] Joe WILKE. *5G Network Architecture and FMC - ITU: Committed*. Juin 2017. URL : https://www.itu.int/en/ITU-T/Workshops-and-Seminars/201707/Documents/Joe-Wilke-%205G%20Network%20Architecture%20and%20FMC.pdf.

[43] Haijun ZHANG, Na LIU, Xiaoli CHU, Keping LONG, A AGHVAMI et Victor LEUNG. « Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges ». In : *IEEE Communications Magazine* 55 (jan. 2017). DOI : 10.1109/MCOM.2017.1600940. URL : https://www.researchgate.net/publication/313611684_Network_Slicing_Based_5G_and_Future_Mobile_Networks_Mobility_Resource_Management_and_Challenges.

[44] Haijun ZHANG, Na LIU, Xiaoli CHU, Keping LONG, Abdol-Hamid AGHVAMI et Victor C. M. LEUNG. « Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges ». In : *CoRR* abs/1704.07038 (2017). arXiv : 1704.07038. URL : http://arxiv.org/abs/1704.07038.

Code source du projet

Configuration du serveur Router Advertisement Daemon

Configuration du serveur Router Advertisement Daemon -

```
# Declarer dans le fichier les directives suivantes et enregistrer
interface s1-eth1 {
  UnicastOnly off;
      AdvSendAdvert on;
      MinRtrAdvInterval 3;
      MaxRtrAdvInterval 10;
  clients {2001:7ab:1:1::a; 2001:7ab:1:1::1;};
  #abro 2001:7ab:1:1::1/64 {
  #      AdvVersionHigh 10;
      #  AdvVersionLow 2;
  #      AdvValidLifeTime 2;
  #   };
  #AdvRASrcAddress { list of IPv6 addresses };
  #ABRO (Authoritative Border Router Option) definitions are of the
     form:
  #abro IPv6-address { list of abro specific options };
      prefix 2001:7ab:1:1::/64 {
            AdvOnLink on;
    # clients {2001:7ab:1::1;};
            AdvAutonomous on;
            AdvRouterAddr off;
      };
};


interface s1-eth2 {
      AdvSendAdvert on;
      MinRtrAdvInterval 3;
      MaxRtrAdvInterval 10;
      prefix 2001:10ab:2::/48 {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};

interface s1-eth3 {
      AdvSendAdvert on;
      MinRtrAdvInterval 3;
      MaxRtrAdvInterval 10;
```

```
      prefix 2001:10ab:3::/48 {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};

interface r1-eth1 {
      AdvSendAdvert on;
      MinRtrAdvInterval 3;
      MaxRtrAdvInterval 10;
      prefix 2001:10ab:1::/48 {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};

#interface s1-eth5 {
#      AdvSendAdvert on; # Le routeur envoie les avertissements
#      prefix 2001:90ab:12::/64
#       {
#            AdvRouterAddr on; # par defaut a off
#            AdvOnLink on;
#            AdvAutonomous on;
#                        # Voir les options dans man radvd.conf
      # On ne met rien d'autre pour l'instant
#       };
#};
```

Code source PYTHON

PYTHON - Code source du programme Mininet

```
#! /usr/bin/env python

import json

import httplib
import os
import subprocess
import time

from mininet.util import irange

from mininet.log import lg
from mininet.log import setLogLevel, info

import ipmininet
from ipmininet.cli import IPCLI
from ipmininet.ipnet import IPNet
from ipmininet.iptopo import IPTopo
from ipmininet.router.config.base import RouterConfig
from ipmininet.router.config.zebra import StaticRoute, Zebra

from mininet.nodelib import LinuxBridge
from mininet.node import Controller, RemoteController, OVSController,
    Host, OVSKernelSwitch


# http://manpages.ubuntu.com/manpages/xenial/man8/ovs-ofctl.8.html

"""
This file contains a simple topology with three routers and three hosts
  a ---- r1 ---- r2 ---- r3 ---- b
                 +
                 c
"""




def addInterfaceToGateway(name):
   data = {
      "interfaces" : [
         {
            "interface-name" : "interface-1",
            "interface-ip" : "2001:90ab:11::3",
            "interface-mask" : "64"
         },
         {
            "interface-name" : "interface-2",
            "interface-ip" : "2001:90ab:21::3",
            "interface-mask" : "64"
         },
```

```
            {
               "interface-name" : "interface-3",
               "interface-ip" : "2001:90ab:31::3",
               "interface-mask" : "64"
            }
         ]
      }
      ret = rest_call('/wm/routing/gateway/' + name, data, 'POST')
      return ret


def addVirtualGateway(name):
      data = {
         "gateway-name" : name,
         "gateway-mac" : "aa:bb:cc:dd:ee:ff"
      }
      ret = rest_call('/wm/routing/gateway', data, 'POST')
      return ret


def addNodePortTupleToGateway(name):
      data = {
         "gateway-name" : name,
         "gateway-ip" : "127.0.0.1",
         "switchports": [
            {
               "dpid": "1",
               "port": "1"
            },
            {
               "dpid": "1",
               "port": "2"
            },
            {
               "dpid": "1",
               "port": "3"
            },

         ]
      }
      ret = rest_call('/wm/routing/gateway/' + name, data, 'POST')
      return ret



# Source from Example Floodlight 1-2
def enableL3Routing():
      data = {
         "enable" : "true"
      }
      ret = rest_call('/wm/routing/config', data, 'POST')
      return ret

def disableL3Routing():
      data = {
         "enable" : "false"
```

```
    }
    ret = rest_call('/wm/routing/config', data, 'POST')
    return ret


def rest_call(path, data, action):
    headers = {
      'Content-type': 'application/json',
      'Accept'   : 'application/json',
    }

    body = json.dumps(data)

    conn = httplib.HTTPConnection('127.0.0.1', 8080)
    conn.request(action, path, body, headers)
    response = conn.getresponse()

    ret = (response.status, response.reason, response.read())
    conn.close()
    return ret

class SimpleTopo(IPTopo):



  def build(self, *args, **kwargs):
    """
    """
    # STATIC ROUTE
    r1_routes = [StaticRoute("::/0", "2001:90ab:11::2")]
    r2_routes = [StaticRoute("::/0", "2001:90ab:21::2")]
    r3_routes = [StaticRoute("::/0", "2001:90ab:31::2")]

    # ADD ROUTER
    r1 = self.addRouter_v6('r1', r1_routes)
    r2 = self.addRouter_v6('r2', r2_routes)
    r3 = self.addRouter_v6('r3', r3_routes)

    # LINK ROUTER
    #self.addLink(r1, r2, params1={"ip": "2001:89ab:12::1/48"},
    #        params2={"ip": "2001:89ab:12::2/48"})

    #self.addLink(r2, r3, params1={"ip": "2001:89ab:23::1/48"},
    #          params2={"ip": "2001:89ab:23::2/48"})

    # ADD SWITCH
    s1 = self.addSwitch( 's1',cls=OVSKernelSwitch, listenPort=6633,
        mac='00:00:00:00:01:01' )

    # LINK SWITCH
    self.addLink(r1, s1,
      params1={"ip": "2001:90ab:11::1/48"},
      params2={"ip": "2001:90ab:11::2/48"})

    self.addLink(r2, s1,
```

```
        params1={"ip": "2001:90ab:21::1/48"},
        params2={"ip": "2001:90ab:21::2/48"})

    self.addLink(r3, s1,
        params1={"ip": "2001:90ab:31::1/48"},
        params2={"ip": "2001:90ab:31::2/48"})

    # LINK HOSTS
    self.addLink(r1, self.addHost('b'),
        params1={"ip": "2001:7ab:1:1::1/64"},
        params2={"ip": "2001:7ab:1:1::a/64"})

    self.addLink(r2, self.addHost('b'),
        params1={"ip": "2001:7ab:2:2::1/64"},
        params2={"ip": "2001:7ab:2:2::b/64"})

    self.addLink(r3, self.addHost('b'),
        params1={"ip": "2001:7ab:3:3::1/64"},
        params2={"ip": "2001:7ab:3:3::c/64"})

    super(SimpleTopo, self).build(*args, **kwargs)

  def addRouter_v6(self, name, staticRoutes):
    return self.addRouter(name, use_v4=False, use_v6=True,
        config=(RouterConfig, {'daemons': [(Zebra, {"static_routes":
        staticRoutes})]}))

ipmininet.DEBUG_FLAG = True
lg.setLogLevel("info")

# Start network*
net = IPNet(topo=SimpleTopo(), use_v4=False, allocate_IPs=False)

try:
    # Add controller - LISTE N ON PORT 6653
    info( '*** Adding Controller 0\n' )
    c0=net.addController(name='c0',controller=RemoteController,
        ip='127.0.0.1:6633', protocol='tcp')

    #
        ********************************************************************************
    # ******   CONTROLLER ENABLE - permet de visualiser les switch -
        hotes      *******
    #
        ********************************************************************************

    #info( '*** Adding Link s1-C0 \n' )
    #net.get('s1').start([c0])


    net.start()


    # CONTROLLER CONFIG
    # c0 sudo ovs-vsctl set bridge br0 protocols=OpenFlow13
    # c0 sudo ovs-vsctl set-controller br0 tcp:0.0.0.0:6633
```

```
# c0 sudo ovs-vsctl set controller br0 connection-mode=out-of-band
# c0 sudo ovs-vsctl list controller



#
    ********************************************************************************
# ******   CONTROLLER DISABLE - switch et hotes non visibles
      *******
#
    ********************************************************************************
# *****           DOIT SE TROUVER avant net.start()
info( '*** Adding Link s1-C0 \n' )
net.get('s1').start([c0])

#a = net.get("a")
b = net.get("b")
#c = net.get("c")

r1 = net.get("r1")
r2 = net.get("r2")
r3 = net.get("r3")

s1 = net.get("s1")

#info( '*** Adding Link s1-c0 \n' )
net.addLink(s1, c0,
       params1={"ip": "2001:90cd:11::1/48"},
       params2={"ip": "2001:90cd:11::2/48"})

# ROUTE S1 - DE RETOUR VERS LE RESEAU INTERNE
#s1.cmd("ip -6 route add 2001:89ab:23::/48 via 2001:90ab:21::1")
#s1.cmd("ip -6 route add 2001:89ab:12::/48 via 2001:90ab:21::1")

s1.cmd("ip -6 route add 2001:7ab:1:1::/64 via 2001:90ab:11::1")
s1.cmd("ip -6 route add 2001:7ab:2:2::/64 via 2001:90ab:21::1")
s1.cmd("ip -6 route add 2001:7ab:3:3::/64 via 2001:90ab:31::1")

# ROUTE S1 - DE RETOUR VERS LE RESEAU INTERNE
#c0.cmd("ip -6 rute add 2001:90ab:21::/48 via 2001:90ab:11::1")
#c0.cmd("ip -6 rute add 2001:89ab:23::/48 via 2001:90ab:11::1")
#c0.cmd("ip -6 rute add 2001:89ab:12::/48 via 2001:90ab:11::1")

#CONFIGURATION CONTROLLER
#c0.cmd("ovs-vsctl add-br mybridge ")
#c0.cmd("ovs-vsctl add-port mybridge eth0 ")

#c0.cmd("ip tuntap add mode tap vport1 ")
#c0.cmd("ip tuntap add mode tap vport2 ")
#c0.cmd("ip tuntap add mode tap vport3 ")

# ADD PORT TO THE BRIDGE
#c0.cmd("ovs-vsctl add-port mybridge r1-eth0")
#c0.cmd("ovs-vsctl add-port mybridge r2-eth1")
#c0.cmd("ovs-vsctl add-port mybridge r3-eth2")
```

```
# ENABLE NETWORK INTERFACE C0-ETH
#c0.cmd("ifconfig r1-eth0 up ")
#c0.cmd("ifconfig r2-eth1 up ")
#c0.cmd("ifconfig r3-eth2 up ")

#Connect to ENS33 Interface
#c0.cmd("ovs-vsctl add-port mybridge ens33")

#c0.cmd("ovs-vsctl add-port mybridge vport1 --add-port mybridge
    vport2 ")
#c0.cmd("ifconfig mybridge 2001:92ef:1::1/48")
#c0.cmd("sudo route add default gw 127.0.0.1 mybridge ")

# CHECK CONFIG
#c0.cmd(" check config ens33 ")
#c0.cmd(" check config mybridge ")

# ADD LINK
#self.addLink(r1, s1,
#params1={"ip": "2001:90ab:11::1/48"},
#params2={"ip": "2001:90ab:11::2/48"})

#self.addLink(r2, s1,
#params1={"ip": "2001:90ab:21::1/48"},
#params2={"ip": "2001:90ab:21::2/48"})

#self.addLink(r3, s1,
#params1={"ip": "2001:90ab:31::1/48"},
#params2={"ip": "2001:90ab:31::2/48"})

# ADD RULES TO ROUTING PACKET
#c0.cmd("sudo ovs-ofctl add-flow mybridge
    dl_type=0x86DD,ipv6_dst=2001:7ab:1::/48,actions=output:1")
#c0.cmd("sudo ovs-ofctl add-flow mybridge
    dl_type=0x86DD,ipv6_dst=2001:7ab:2::/48,actions=output:2")
#c0.cmd("sudo ovs-ofctl add-flow mybridge
    dl_type=0x86DD,ipv6_dst=2001:7ab:3::/48,actions=output:3")

c0.cmd("ip -6 route add 2001:7ab:1:1::/64 via 2001:90ab:11::1")
c0.cmd("ip -6 route add 2001:7ab:2:2::/64 via 2001:90ab:11::1")
c0.cmd("ip -6 route add 2001:7ab:3:3::/64 via 2001:90ab:11::1")


# http://www.delafond.org/traducmanfr/man/man8/ip6tables.8.html

# GATEWAY FOR R1
#s1.cmd('ip -6 route add ::/0 via 2001:90cd:11::2')

# FOR Routing Packet
#
# ACCEPT ICMP REDIRECT MESSAGES
s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/accept_redirects")
#
# SEND ICMP REDIRECT MESSAGES
s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/send_redirects")
#
```

```
    # SOURCE ROUTED PACKETS
    s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/accept_source_route")

      # ROUTER 1
    #
        ************************************************************ROUTER
        1 - FORWARDING
        ********************************************************************

    s1.cmd(" echo 1 >/proc/sys/net/ipv6/conf/all/forwarding")

    s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/s1-eth1/forwarding")
    s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/s1-eth2/forwarding")
    s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/s1-eth3/forwarding")
    s1.cmd("echo 1 > /proc/sys/net/ipv6/conf/s1-eth4/forwarding")
    #
        ************************************************************ROUTER
        2 - FORWARDING FROM R2-ETH2 TO R2-ETH3
        ********************************************************************
    # ------- NETTOYAGE TABLES IPv6
    s1.cmd("ip6tables -F")

  # ************************************************************ROUTER
     2 - FORWARDING
     ********************************************************************

    #envoi d un ping
    s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
        echo-request -j ACCEPT")
    s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
        echo-request -j ACCEPT ")
    s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
        echo-request -j ACCEPT ")
    #reponse aux ping
    s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
        echo-reply -j ACCEPT")
    s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
        echo-reply -j ACCEPT")
    s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
        echo-reply -j ACCEPT")

    #neighbour solicitation
    s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
        neighbour-solicitation -j ACCEPT")
    s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
        neighbour-solicitation -j ACCEPT")
    s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
        neighbour-solicitation -j ACCEPT")

    #neighbour advertisement
    s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
        neighbour-advertisement -j ACCEPT ")
    s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
        neighbour-advertisement -j ACCEPT")
    s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
        neighbour-advertisement -j ACCEPT")
```

```
#router solicitation
s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")

#router advertisement
s1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
s1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
s1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")

# Routing Header (58)
s1.cmd(" ip6tables -A INPUT -m ipv6header --header 43-j ACCEPT")
s1.cmd(" ip6tables -A OUTPUT -m ipv6header --header 43 -j ACCEPT")
s1.cmd(" ip6tables -A FORWARD -m ipv6header --header 43 -j
    ACCEPT")


# ROUTER 1
#
    ************************************************************ROUTER
    1 - FORWARDING
    ****************************************************************

r1.cmd(" echo 1 >/proc/sys/net/ipv6/conf/all/forwarding")
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/r1-eth0/forwarding")
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/r1-eth1/forwarding")

# GATEWAY FOR R1
r1.cmd('ip -6 route add ::/0 via 2001:90ab:11::2')

# GATEWAY FOR B
r1.cmd("ip -6 route add 2001:7ab:1:1::/64 dev r1-eth1")

#   --- Enable Accept router advertissement
#r1.cmd("echo 2 > /proc/sys/net/ipv6/conf/r1-eth0/accept_ra")
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/r1-eth0accept_ra_pinfo")
r1.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r1-eth0/accept_redirects")

#r1.cmd("echo 2 > /proc/sys/net/ipv6/conf/r1-eth1/accept_ra")
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/r1-eth1/accept_ra_pinfo")
r1.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r1-eth1/accept_redirects")


# FOR Routing Packet
#
# ACCEPT ICMP REDIRECT MESSAGES
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/accept_redirects")
```

```
#
# SEND ICMP REDIRECT MESSAGES
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/send_redirects")
#
# SOURCE ROUTED PACKETS
r1.cmd("echo 1 > /proc/sys/net/ipv6/conf/all/accept_source_route")

#
    ************************************************************ROUTER
    1- FORWARDING FROM R2-ETH2 TO R2-ETH3
    ************************************************************
# ------- NETTOYAGE TABLES IPv6
r1.cmd("ip6tables -F")

#s Allow traffic for link-local addresses:
r1.cmd("ip6tables -A INPUT -s fe80::/10 -j ACCEPT")

# Normally, link-local packets should NOT be forwarded and don't
    need an
# entry in the FORWARD rule.
# However, when bridging in Linux (e.g. in Xen or OpenWRT), the
    FORWARD rule is
# needed:

r1.cmd("ip6tables -A FORWARD -s fe80::/10 -j ACCEPT")
r1.cmd("ip6tables -A OUTPUT -s fe80::/10 -j ACCEPT")

r1.cmd("ip6tables -A INPUT -p icmpv6 -j ACCEPT")
r1.cmd("ip6tables -A FORWARD -p icmpv6 -j ACCEPT")
r1.cmd("ip6tables -A OUTPUT -p icmpv6 -j ACCEPT")

r1.cmd("ip6tables -A REDIRECT -j ACCEPT")

#envoi d un ping
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT")
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
#reponse aux ping
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")

#neighbour solicitation
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
```

```
#neighbour advertisement
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT ")
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")

#router solicitation
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")

#router advertisement
r1.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r1.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r1.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")

# Routing Header (58)
r1.cmd(" ip6tables -A INPUT -m ipv6header --header 43-j ACCEPT")
r1.cmd(" ip6tables -A OUTPUT -m ipv6header --header 43 -j ACCEPT")
r1.cmd(" ip6tables -A FORWARD -m ipv6header --header 43 -j
    ACCEPT")


# ROUTER 2
# FORWARDING ROUTER ADVERTISSEMENT
#   --- Enable Forwarding
r2.cmd(" echo 1 >/proc/sys/net/ipv6/conf/all/forwarding")
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth0/forwarding")
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth1/forwarding")


#   --- Enable Accept router advertissement
r2.cmd("echo 2 > /proc/sys/net/ipv6/conf/r2-eth0/accept_ra")
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth0accept_ra_pinfo")
r2.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r2-eth0/accept_redirects")

r2.cmd("echo 2 > /proc/sys/net/ipv6/conf/r2-eth1/accept_ra")
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth1/accept_ra_pinfo")
r2.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r2-eth1/accept_redirects")

r2.cmd("echo 2 > /proc/sys/net/ipv6/conf/r2-eth2/accept_ra")
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth2/accept_ra_pinfo")
r2.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r2-eth2/accept_redirects")

r2.cmd("echo 2 > /proc/sys/net/ipv6/conf/r2-eth3/accept_ra")
```

```
r2.cmd("echo 1 > /proc/sys/net/ipv6/conf/r2-eth3/accept_ra_pinfo")
r2.cmd("echo 1 >
    /proc/sys/net/ipv6/conf/r2-eth3/accept_redirects")
#net.bridge.bridge-nf-call-iptables=0

#
    ************************************************************ROUTER
    2 - FORWARDING FROM R2-ETH2 TO R2-ETH3
    ************************************************************
# ------- NETTOYAGE TABLES IPv6
r2.cmd("ip6tables -F")


# GATEWAY FOR R2
r2.cmd('ip -6 route add ::/0 via 2001:90ab:21::2')

# GATEWAY FOR B
r2.cmd("ip -6 route add 2001:7ab:2:2::/64 dev r2-eth1")

  #envoi d un ping
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT")
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
#reponse aux ping
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")

#neighbour solicitation
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")

#neighbour advertisement
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT ")
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")

#router solicitation
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
```

```
    router-solicitation -j ACCEPT ")

#router advertisement
r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")

# Router Solicitation / Advertisement
#r2.cmd("ip6tables -I INPUT -p icmpv6 --icmpv6-type 133/0 -j
    ACCEPT")
#r2.cmd("ip6tables -A INPUT -p icmpv6 -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -p all -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p tcp -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p icmp -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p udp -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p ah -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p esp -j ACCEPT")

#r2.cmd("ip6tables -I INPUT -p icmpv6 -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -p icmpv6 -j ACCEPT")

#r2.cmd("ip6tables -A INPUT -m conntrack --ctstate RELATED,
    INVALID, ESTABLISHED -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -m conntrack --ctstate RELATED,
    INVALID, ESTABLISHED -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -m conntrack --ctstate RELATED,
    INVALID, ESTABLISHED -j ACCEPT")

#r2.cmd("ip6tables -A INPUT -t nat -i r2-eth2 -j ACCEPT")
#r2.cmd("ip6tables -A OUTPUT -t nat -o r2-eth3 -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -i r2-eth2 -o r2-eth3 -m state
    --state INVALID, RELATED,ESTABLISHED -j ACCEPT ")

#FORWARD
#r2.cmd("ip6tables -A FORWARD -m state --state
    ESTABLISHED,RELATED -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -i r2-eth3 -m state --state
    ESTABLISHED,RELATED -j ACCEPT")
# sudo iptables -A FORWARD --in-interface eth0 -j ACCEPT
#r2.cmd("ip6tables -t filter -A FORWARD -i r2-eth0 -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -i r2-eth3 -o r2-eth2 -m state
    --state NEW, ESTABLISHED,RELATED -j ACCEPT")
#r2.cmd("ip6tables -A FORWARD -i r2-eth2 -o r2-eth3 -j ACCEPT")
#r2.cmd("ip6tables -t filter -A FORWARD-i r2-eth2 -j ACCEPT")
#r2.cmd("ip6tables -t filter -A FORWARD -i r2-eth3 -j ACCEPT")

#PRE ROUTING
#r2.cmd("ip6tables -t nat -I PREROUTING -p icmpv6 -m icmpv6 -j
    REDIRECT ")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p all -j DNAT
    --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p icmp -j
    DNAT --to-destination 2001:7ab:2::b")
```

```
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p tcp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p udp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p ah -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p esp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING -i r2-eth2 -p icmpv6 -j
   DNAT --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A PREROUTING ")
#r2.cmd("ip6tables -t nat -A PREROUTING -p icmpv6 ")
#r2.cmd("ip6tables -t nat -A PREROUTING -d ff02::1 -j DNAT
   --to-destination 2001:7ab:2::b")

# PREROUTING - OUTPUT
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p all -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p icmp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p tcp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p udp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p ah -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p esp -j DNAT
   --to-destination 2001:7ab:2::b")
#r2.cmd("ip6tables -t nat -A INPUT -i r2-eth2 -p icmpv6 -j DNAT
   --to-destination 2001:7ab:2::b")

#POSTROUTING
#r2.cmd("ip6ptables -t nat -A POSTROUTING -j MASQUERADE")
#r2.cmd("ip6tables -t nat -A POSTROUTING -o r2-eth3 -j
   MASQUERADE")
#r2.cmd("ip6tables -t nat -A POSTROUTING -o r2-eth3 -j
   MASQUERADE")
#r2.cmd("ip6tables -t nat -A POSTROUTING -i r2-eth2 -j
   MASQUERADE")
#r2.cmd("ip6tables -t nat -A POSTROUTING -i r2-eth3 -j
   MASQUERADE")
#ip6tables -t nat -A POSTROUTING -d 2.2.2.3 -j SNAT --to 1.1.1.3
# ip6tables -t nat -I PREROUTING -i eth0 -p tcp -m tcp --dport 22
   -j REDIRECT --to-ports 2222
# sudo iptables -t nat -A POSTROUTING --out-interface eth1 -j
   MASQUERADE
#r2.cmd("ip6tables -A REDIRECT -i r2-eth3 -o r2-eth2 -icmp-type
   icmpv6 -m state --state ESTABLISHED,RELATED -j ACCEPT")

#envoi d un ping
#r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
   echo-request -j ACCEPT ")
#r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
   echo-request -j ACCEPT")
#r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
   echo-request -j ACCEPT ")
```

```
#reponse aux ping
#r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
   echo-reply -j ACCEPT")
#r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
   echo-reply -j ACCEPT")
#r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
   echo-reply -j ACCEPT")

#neighbour solicitation
#r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
   neighbour-solicitation -j ACCEPT")
#r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
   neighbour-solicitation -j ACCEPT")
#r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
   neighbour-solicitation -j ACCEPT")

#neighbour advertisement
#r2.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
   neighbour-advertisement -j ACCEPT")
#r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
   neighbour-advertisement -j ACCEPT ")
#r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
   neighbour-advertisement -j ACCEPT")

#router solicitation
#r2.cmd("ip6tables -t filter -A INPUT -i r2-eth0 -p icmpv6
   --icmpv6-type router-solicitation -j ACCEPT ")
#r2.cmd("ip6tables -t filter -A INPUT -i r2-eth1 -p icmpv6
   --icmpv6-type router-solicitation -j ACCEPT ")
#r2.cmd("ip6tables -t filter -A INPUT -i r2-eth3 -p icmpv6
   --icmpv6-type router-solicitation -j ACCEPT ")

#r2.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
   router-solicitation -j ACCEPT ")
#r2.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
   router-solicitation -j ACCEPT ")

#router advertisement
#r2.cmd("ip6tables -A INPUT -p icmpv6 --icmpv6-type
   router-advertisement -i r2-eth2 -d ff02::1 -j ACCEPT ")
#r2.cmd("ip6tables -A INPUT -p icmpv6 --icmpv6-type
   router-advertisement -o r2-eth3 -d ff02::1 -j ACCEPT ")
#r2.cmd("ip6tables -A OUTPUT -p icmpv6 --icmpv6-type
   router-advertisement -d ff02::1 -j ACCEPT ")
#r2.cmd("ip6tables -A FORWARD -p icmpv6 --icmpv6-type
   router-advertisement -d ff02::1 -j ACCEPT ")

r3.cmd("echo 1 > /proc/sys/net/ipv6/ip_forward")

# GATEWAY FOR R3
r3.cmd('ip -6 route add ::/0 via 2001:90ab:31::2')

# GATEWAY FOR B
r3.cmd("ip -6 route add 2001:7ab:3:3::/64 dev r3-eth1")

#
```

```
    ************************************************************ROUTER
    2 - FORWARDING FROM R2-ETH2 TO R2-ETH3
    ************************************************************
# ------- NETTOYAGE TABLES IPv6
r3.cmd("ip6tables -F")
#
    ************************************************************ROUTER
    2 - FORWARDING
    ************************************************************

# ROUTER 3
# ------- NETTOYAGE TABLES IPv6
#r3.cmd("ip6tables -F")
#r3.cmd("ip6tables -t filter -A FORWARD -i r1-eth1 -j ACCEPT")
#r3.cmd("ip6tables -t filter -A FORWARD -i r1-eth0 -j ACCEPT")


#envoi d un ping
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT")
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
#reponse aux ping
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")

#neighbour solicitation
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")

#neighbour advertisement
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT ")
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")

#router solicitation
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
```

```
#router advertisement
r3.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r3.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
r3.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")



# HOST B
#   --- Enable Accept router advertissement
b.cmd("echo 2 > /proc/sys/net/ipv6/conf/b-eth0/accept_ra")
b.cmd("echo 1 > /proc/sys/net/ipv6/conf/b-eth0accept_ra_pinfo")
b.cmd("echo 1 > /proc/sys/net/ipv6/conf/b-eth0/accept_redirects")


# ------- NETTOYAGE TABLES IPv6
b.cmd("ip6tables -F")

# Allow traffic for link-local addresses: FE80 with RA
b.cmd("ip6tables -A INPUT -s fe80::/10 -j ACCEPT")

# Normally, link-local packets should NOT be forwarded and don't
    need an
# entry in the FORWARD rule.
# However, when bridging in Linux (e.g. in Xen or OpenWRT), the
    FORWARD rule is
# needed:
b.cmd("ip6tables -A FORWARD -s fe80::/10 -j ACCEPT")
b.cmd("ip6tables -A OUTPUT -s fe80::/10 -j ACCEPT")

b.cmd("ip6tables -A INPUT -p icmpv6 -j ACCEPT")
b.cmd("ip6tables -A FORWARD -p icmpv6 -j ACCEPT")
b.cmd("ip6tables -A OUTPUT -p icmpv6 -j ACCEPT")

b.cmd("ip6tables -A REDIRECT -j ACCEPT")

#envoi d un ping
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT")
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-request -j ACCEPT ")
#reponse aux ping
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    echo-reply -j ACCEPT")

#neighbour solicitation
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
```

```
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT")

#neighbour advertisement
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT ")
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT")

#router solicitation
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT ")

#router advertisement
b.cmd("ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
b.cmd("ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")
b.cmd("ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT ")



# XTERM TERMINAL
# B - Xterm pour running program Switching Node
b.cmdPrint("xterm -name 'B - Switching Node' -fg blue -bg white
    -geometry 95x42+0+0 -fa 'Monospace' -fs 10w -e
    './topo-memoire-H2-script.py ; $SHELL' & ")

# B - Xterm for ping and check connectivity
b.cmdPrint("xterm -name 'B - Ping ' -fg blue -bg white -geometry
    78x18+0-78 -fa 'Monospace' -fs 10w -e
    './topo-memoire-H2-PING-script.py ; $SHELL' & ")

# SWITCH S1
s1.cmdPrint("xterm -name 'SWITCH 1' -fg red -bg white -geometry
    95x27+0-0 -fa 'Monospace' -fs 10w -e
    './topo-memoire-S1-script.py ; $SHELL' & ")

# ROUTER R1 R2 R 3
r1.cmdPrint("xterm -name 'ROUTER 1' -fg red -bg white -geometry
    122x15+700+0 -fa 'Monospace' -fs 10w -e
    './topo-memoire-R1-script.py ; $SHELL' & ")
r2.cmdPrint("xterm -name 'ROUTER 2' -fg DarkOrange3 -bg white
    -geometry 122x15+700+330 -fa 'Monospace' -fs 10w -e
    './topo-memoire-R2-script.py ; $SHELL' & ")
r3.cmdPrint("xterm -name 'ROUTER 3' -fg purple -bg white
    -geometry 122x15+700+630 -fa 'Monospace' -fs 10w -e
```

```
        './topo-memoire-R3-script.py ; $SHELL' & ")

    IPCLI(net)
finally:
  net.stop()
```

PYTHON - Code source du terminal XTERM HOST 2

```
#GET PID of XTerm

myBackgroundXtermPID=$$
echo "********************** HOST 2 - PID : $myBackgroundXtermPID *
    ********************"

echo 2 > /proc/sys/net/ipv6/conf/b-eth0/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/b-eth0/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/b-eth0/autoconf
echo 1 > /proc/sys/net/ipv6/conf/b-eth0/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp

echo 2 > /proc/sys/net/ipv6/conf/b-eth1/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/b-eth1/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/b-eth1/autoconf
echo 1 > /proc/sys/net/ipv6/conf/b-eth1/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp

echo 2 > /proc/sys/net/ipv6/conf/b-eth2/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/b-eth2/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/b-eth2/autoconf
echo 1 > /proc/sys/net/ipv6/conf/b-eth2/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp


#conserve les connections en cours
ip6tables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
ip6tables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
ip6tables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

#ip6tables -A INPUT -p icmp --icmp-type 8 -m state --state
    NEW,ESTABLISHED,RELATED -j ACCEPT
#ip6tables -A OUTPUT -p icmp --icmp-type 0 -m state --state
    ESTABLISHED,RELATED -j ACCEPT

#ip6tables -A OUTPUT -p icmp --icmp-type 8 -m state --state
    NEW,ESTABLISHED,RELATED -j ACCEPT
#ip6tables -A INPUT -p icmp --icmp-type 0 -m state --state
    ESTABLISHED,RELATED -j ACCEPT

#envoi d un ping
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type echo-request -j
    ACCEPT
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT

#reponse aux ping
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type echo-request -j
    ACCEPT
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT

#reponse aux ping
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type echo-request -j
```

```
    ACCEPT
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT

#neighbour solicitation - advertisement
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT


ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT

ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT

#router solicitation - advertisement
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT

ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT

ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-solicitation -j ACCEPT
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type
    router-advertisement -j ACCEPT
# Allow others ICMPv6 types but only if the hop limit field is 255.

ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type redirect -j ACCEPT
ip6tables -A FORWARD -p icmpv6 --icmpv6-type redirect -j ACCEPT


# Mise a zero du ficher d echange originale - Adresse IPv6 source =
    B-ETH0
sed -i 's/export DB_INTERF_PREC=.*".*"/export DB_INTERF_PREC=""/g'
    /etc/profile.d/topo-env.sh
sed -i 's/export DB_IPV6_PREC=.*".*"/export DB_IPV6_PREC=""/g'
    /etc/profile.d/topo-env.sh

sed -i 's/export DB_INTERF=.*".*"/export DB_INTERF="b-eth0"/g'
    /etc/profile.d/topo-env.sh
sed -i 's/export DB_IPV6=.*".*"/export DB_IPV6="2001:7ab:1::a"/g'
    /etc/profile.d/topo-env.sh && source /etc/profile.d/topo-env.sh &&
    echo $DB_IPV6
cp /etc/profile.d/topo-env.sh /etc/profile.d/topo-env-dup.sh
```

```
# java -jar
   /home/administrateur/mininet/IdeaProjects/SSC-Mode3/out/artifacts/SSC_Mode3_jar/SS

#kill -HUP $myBackgroundXtermPID
#exec exit
```

PYTHON - Code source du terminal XTERM SWITCH 1

```
#GET PID of XTerm
myBackgroundXtermPID=$$
echo "********************** SWITCH 1 - PID : $myBackgroundXtermPID
    * ********************"

#envoi d un ping
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type echo-request -j
    ACCEPT
ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT

#reponse aux ping
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type echo-request -j
    ACCEPT
ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT


#reponse aux ping
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type echo-request -j
    ACCEPT
ip6tables -t filter -A FORWARD -p icmpv6 --icmpv6-type echo-reply -j
    ACCEPT

#neighbour solicitation - advertisement
#ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -m hl --hl-eq 255 -j ACCEPT
i#p6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -m hl --hl-eq 255 -j ACCEPT
#ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-solicitation -j ACCEPT
#ip6tables -t filter -A OUTPUT -p icmpv6 --icmpv6-type
    neighbour-advertisement -j ACCEPT

#router solicitation - advertisement
#ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-solicitation -m hl --hl-eq 255 -j ACCEPT
#ip6tables -t filter -A INPUT -p icmpv6 --icmpv6-type
    router-advertisement -m hl --hl-eq 255 -j ACCEPT


# Allow others ICMPv6 types but only if the hop limit field is 255.

#ip6tables -A INPUT -p icmpv6 --icmpv6-type router-advertisement -m hl
    --hl-eq 255 -j ACCEPT
#ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -m
    hl --hl-eq 255 -j ACCEPT
#ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -m
    hl --hl-eq 255 -j ACCEPT
#ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -m hl --hl-eq 255
    -j ACCEPT

# Limit most NDP messages to the local network.
```

```
#ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-solicitation -m
    hl --hl-eq 255 -j ACCEPT
#ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-advertisement
    -m hl --hl-eq 255 -j ACCEPT
#ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-solicitation -m hl
    --hl-eq 255 -j ACCEPT

echo 2 > /proc/sys/net/ipv6/conf/s1-eth1/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/s1-eth1/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/s1-eth1/autoconf
echo 1 > /proc/sys/net/ipv6/conf/s1-eth1/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp

echo 2 > /proc/sys/net/ipv6/conf/s1-eth2/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/s1-eth2/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/s1-eth2/autoconf
echo 1 > /proc/sys/net/ipv6/conf/s1-eth2/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp

echo 2 > /proc/sys/net/ipv6/conf/s1-eth3/accept_ra
echo 1 > /proc/sys/net/ipv6/conf/s1-eth3/accept_ra_pinfo
echo 1 > /proc/sys/net/ipv6/conf/s1-eth3/autoconf
echo 1 > /proc/sys/net/ipv6/conf/s1-eth3/accept_redirects
echo 1 >/proc/sys/net/ipv6/conf/all/proxy_ndp

#SHOW Rules OpenFlow
sudo ovs-ofctl show mybridge


#SHOW Rules OpenFlow
sudo ovs-ofctl dump-flows mybridge


echo '********************** Mise en suspend - 5 secondes *
    ********************'
kill -HUP $myBackgroundXtermPID
exec exit
```

PYTHON - Code source du terminal XTERM ROUTER 1

```
#GET PID of XTerm
myBackgroundXtermPID=$$
echo "********************** ROUTEUR 1 - PID : $myBackgroundXtermPID
    ********************"

# Running TCPDUMP
tcpdump
```

PYTHON - Code source du terminal XTERM ROUTER 2

```
#GET PID of XTerm
myBackgroundXtermPID=$$
echo "********************** ROUTEUR 2 - PID : $myBackgroundXtermPID
    ********************"

# Running TCPDUMP
tcpdump
```

PYTHON - Code source du terminal XTERM ROUTER 3

```
#GET PID of XTerm
myBackgroundXtermPID=$$
echo "********************** ROUTEUR 3 - PID : $myBackgroundXtermPID
    ********************"

# Running TCPDUMP
tcpdump
```

PYTHON - Scapy - Code source du programme Ping-Scapy

```
#! /usr/bin/env python
# -*- encoding: utf-8 -*-

from sys import argv,exit
from os import path
from scapy.all import *
import re
import time

import os
import subprocess
import subprocess as sp
import os, subprocess
from subprocess import call

# Active le support de IPv6
from scapy.layers.inet6 import IP
from scapy.config import conf
conf.ipv6_enabled = True
from scapy.all import *

# Active les erreur de SCAPY
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *

# source:
    https://github.com/secdev/scapy/blob/master/scapy/layers/inet6.py

def bash_command(cmd):
    subprocess.Popen(['/bin/bash', '-c', cmd])

#
    ********************************************************************************
# ****        FUNTION check : recupere value for a specific key
    ****
#
    ********************************************************************************
def check(key):

   # Fichier topo-env-dup mis a jour a chaque changement au niveau des
      interface
   #
   # Le fichier est remplace par la version originale se trouvant dans
      /etc/profile.d/topo-env.sh

   datafile = file('/etc/profile.d/topo-env-dup.sh')
    #found = False #this isn't really necessary
   for line in datafile:
     if key in line:
          #found = True #not necessary
        return line
   return ""
```

```
#
    ********************************************************************************
# ****        FUNTION extract : extract value from a string       *****
#
    ********************************************************************************
def extract(value, text):

  try:

     m = re.search('\"(.*)\"', text)
     #print "Value finded : %s" %(m.group(1))
     return m.group(1)

  except AttributeError:

     return ''


#
    ********************************************************************************
# ****        FUNCTION Ping mode Scapy effectue un ping a l'aide de
    SCAPY ******
#
    ********************************************************************************
def pingbis(mac, interf, ipv6src, ipv6prec, interfaceprec, str_time,
    numseq):

     try:

        # IPv6 of Google IPv6 DNS
        ipv6dest ="2001:4860:4860::8888"

        #print mac
        eth = Ether()
        if mac =="":
          mac="66:C8:1A:53:4E:A9"

        eth.src = mac
        eth.type = 0x8100

        #print "IP"
        ip = IPv6()
        ip.src = ipv6src
        ip.dst = ipv6dest
        ip.hlim = 255

        #print "ICMP"
        # Generation d'' ID ICMP aleatoire
        icmpid=random.randrange(0,65535,1)
        icmp =ICMPv6EchoRequest(seq=numseq, id=icmpid)
        icmp.data = "memoire-2018-2019"

        #print "Route"
        route=[]
        #route.append(ipv6src)
```

```
route.append(ipv6dest)

route_1= [ '2001:7ab:1:1::1', '2001:90ab:11:1::1']
route_2= [ '2001:90ab:21::1']
route_3= [ '2001:90ab:31::1']

#print "IPv6 ext"
ipv6extHR = IPv6ExtHdrRouting()
ipv6extHR.addresses = route
ipv6extHR.type = 2
ipv6extHR.len = 1

#print "Packet"
packet= ip / ipv6extHR / icmp

# Build and recalculate the whole packet
packet = Ether(bytes(packet))


# ************************** INFO
   ***************************************
# send() pour envoyer un paquet de la couche 3 ;
# sendp() pour envoyer un paquet de la couche 2 ;

# sr1() pour envoyer un paquet de la couche 3 et retourner la
   premiere reponse ;
# srp1() pour envoyer un paquet de la couche 2 et retourner la
   premiere reponse ;

# sr() pour envoyer un paquet de la couche 3 et retourner
   toutes les reponses ;
# srp() pour envoyer un paquet de la couche 2 et retourner
   toutes les reponses.

ip_packet=IPv6(dst=ipv6dest,hlim=255, nh=0)

# Ajouter IPv6 source quand on change de noeud
ext_1=IPv6ExtHdrHopByHop(nh=60)
ext_2=IPv6ExtHdrDestOpt(nh=43)
ext_3=IPv6ExtHdrRouting( type=253, nh=60 )
#ext_3=IPv6ExtHdrRouting( nh=60 )
ext_4=IPv6ExtHdrDestOpt(nh=58) # nh=59 == No Next Header

icmp_packet= ICMPv6EchoRequest( seq=numseq, id=icmpid, data="
   UNamur 2019")

final_packet=ip_packet/ext_1/ext_2/ext_3/ ext_4 /icmp_packet

if interf=="b-eth0":
  #final_packet = IPv6(dst=ipv6dest, hlim=255, nh=58) /
     ICMPv6EchoRequest(seq=numseq, id=icmpid, data=" UNamur
     2019")

  #final_packet.show()
  a = send(final_packet ,iface=interf, verbose=1)
```

```
        else:
          #if interf=="b-eth1":
          # route= route_1
          #else:
          #route= route_2

          #final_packet.show()
          a = send(final_packet ,iface=interf, verbose=1)

        print "N Seq : %s - Mac : %s - Interface : %s - Adr IPV6 src :
            %s - Adr IPV6 dest : %s - Interf Prec: %s - Adr IPV6 prec :
            %s " %(numseq, mac, interf, ipv6src, ipv6dest,interfaceprec,
            ipv6prec)



        return 0
      except Exception as e:
        results = None
        print e
        return results

      #    except AttributeError:
      #
      #    return ''-1 # apply your error handling

# Num sequence of Echo request
sequence =0

while True:

  subprocess.call("clear")
  sequence = sequence +1

  mac = check("DB_MAC")
  mac = extract("DB_MAC",mac)

  mac= mac.replace('-', ':')

  ipv6 = check("DB_IPV6")
  ipv6 = extract("DB_IPV6",ipv6)

  interface = check("DB_INTERF")
  interface = extract("DB_INTERF",interface)

  ipv6prec= check("DB_IPV6_PREC")
  ipv6prec = extract("DB_IPV6_PREC",ipv6prec)

  interfaceprec = check("DB_INTERF_PREC")
  interfaceprec = extract("DB_INTERF_PREC",interfaceprec)

  str_time = check("DB_TIME")
  str_time = extract("DB_TIME",str_time)

  # Refresh route from Kernel to Scapy
  conf.iface6
```

```
  conf.route6.resync()

  print conf.route6
  if ipv6 != "" :
    pingbis(mac, interface, ipv6, ipv6prec, interfaceprec, str_time,
        sequence)

  time.sleep(1)
```

Code source JAVA

JAVA - Code source du programme SSC-Mode2

```
import com.sun.xml.internal.ws.util.StringUtils;
import sun.awt.geom.AreaOp;
import sun.nio.ch.Net;

import java.io.*;
import java.lang.reflect.Field;
import java.time.Duration;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.*;

import java.io.IOException;
import java.io.InputStream;
import java.net.*;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.*;

import static java.lang.System.out;
import static java.lang.Thread.sleep;
import static java.time.LocalTime.now;

import java.net.Inet4Address;
import java.net.Inet6Address;
import java.net.InetAddress;
import java.net.UnknownHostException;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Random;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class GetInfo {

   public static void main(String args[]) throws SocketException,
       IOException, InterruptedException {

      boolean process = true ;
      boolean show =false;

      String cmdnet="";
      int rndNumber = 0 ;
      int delai =500; // Delai avant rafraichissement - en milliseconde
      int delaiChange =500; // Delai lors du change avec
         rafraichissement - en milliseconde
      int delaiSwitching =10; // Delai entre changement entre 2 noeud
         - en seconde
      boolean change =false ;
```

```
// TEST Si le tableu est vide == 0 interface


/*****************************************************
 *          PART INITZIALIZATION Structure *
 *****************************************************/

cmdnet = "clear ";
runSystemCommand(cmdnet, "");

//Instantiate structure to savd information
out.printf("*****************************************************\n");
out.printf("***     Starting application....    ***\n");
out.printf("*****************************************************\n");


// Nice to Have: recuperer le nom d hote pour l associer au nom
   de l'interface

//out.printf("DEL Default Route \n\n");
//cmdnet="ip -6 route del ::/0 ";
//runSystemCommand(cmdnet, "");

// cmdnet="ip -6 route del default ";
// runSystemCommand(cmdnet, "");

out.printf("Disable Interface - B-ETH1 \n");
cmdnet = "ifconfig ";
runSystemCommand(cmdnet, "b-eth1 down ");


out.printf("Disable Interface - B-ETH2 \n\n");
cmdnet = "ifconfig ";
runSystemCommand(cmdnet, "b-eth2 down ");


 out.printf("ADD Default Route - B-ETH0 \n\n");
//cmdnet="ip -6 route add ::/0 via 2001:7ab:1::1 ";
//runSystemCommand(cmdnet, "");

cmdnet="ip -6 route add default via 2001:7ab:1:1:0:0:0:1 ";
runSystemCommand(cmdnet, "");

out.printf("Initializing Variable environmement - Key DB_MAC\n");
executeCommand("DB_MAC", "");

//MODIFY DB_INTERFACE
out.printf("Initializing Variable environmement - Key
   DB_INTERF\n");
executeCommand("DB_INTERF", "b-eth0");

out.printf("Initializing Variable environmement - Key
   DB_IPV6\n");
//executeCommand("DB_IPV6", "2001:7ab:1:1:0:0:0:a");
```

```
out.printf("Initializing Variable environmement - Key
    DB_IPv6_PREC\n");
executeCommand("DB_IPV6_PREC", "");

//MODIFY DB_INTERFACE
out.printf("Initializing Variable environmement - Key
    DB_INTERF_PREC\n");
executeCommand("DB_INTERF_PREC", "");

//MODIFY DB_INTERFACE

DateTimeFormatter dtfinit =
    DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
LocalDateTime nowinit = LocalDateTime.now();
String timechginit = dtfinit.format(nowinit);
out.printf("Initiazing Variable environmement - Key DB_TIME\n");
executeCommand("DB_TIME", timechginit);

// Duplicate file
cmdnet = " cp /etc/profile.d/topo-env.sh
    /etc/profile.d/topo-env-dup.sh ";
runSystemCommand(cmdnet, "");


ArrayList<NetInfo> NetArray = new ArrayList<NetInfo>();
NetArray= initArrayNet(NetArray);
out.printf("\nStructure NetInfo initialized" +"\n\n");

 /*************************************************************************
 *          PART MONITORING CHANGE ON NETWORK INTERFACERS *
 *************************************************************************/
while (process == true ){
   if (show==true){
      out.printf("*****************************************************\n");
      out.printf("***  Informations Interfaces....   ***\n");
      out.printf("*****************************************************\n");
   }

    /*****************************************
    *          PART INITZIALIZATION *
    *****************************************/
   // Retrieve information about interfaces
   Enumeration<NetworkInterface> nets =
      NetworkInterface.getNetworkInterfaces();

   //Generate random number
   Random rand = new Random();
   rndNumber= rand.nextInt(50) + 1;

   //Enumeration<NetworkInterface> netscount =
      NetworkInterface.getNetworkInterfaces();
   //out.printf("InetAddress found : %s\n\n",
      Collections.list(netscount).size());
   int NetIntcount =0;
   int idx = 0;
```

```
//Initialization & Interate for each interface retrievd
for (NetworkInterface netint : Collections.list(nets)) {
   NetInfo ni = new NetInfo();

   //Initialization of idx
   idx=0;

   ni= displayInterfaceInformation(netint, ni,true );
   String DisplayName =
      netint.getDisplayName().toString().replace(" ","");


   if (!DisplayName.equals("lo")) { // IF Interface is not
      loopback

      /*******************************************************
       *          PART SEARCHING INTERFACES     *
       *******************************************************/

      idx = searchInterface(NetArray, DisplayName);
      out.printf("Searching interface : " + DisplayName +"
         (idx=" + idx +") is existing : " + "\n");

      // SI l interface a ete trouvee et est existante
      if (idx != -1) {

         // Mise a jour de l'interface comme toujours
            existante dans la configuration actuelle
         NetArray.get(idx).setInetTime(now());
         NetArray.get(idx).setCheck(rndNumber);

         String ip= "";
         ip = findIPv6(netint, "2001");

       // Save adresse IPv6 si pas encore presente
       if (!ip.equals("")){

            // TESTER IPv6 dan le fichier de configuration si
               presente sinon MAJ du fichier avec IP
            NetArray.get(idx).setadrIP(ip);

            // MODIFY DB_MAC
            String mac ="";
            mac= NetArray.get(idx).GetMac();

            out.printf(" * Modification Variable
               environmement - Key DB_MAC : " + mac + "\n");
            executeCommand("DB_MAC", mac);



            String
               interf_act=searchLine("/etc/profile.d/topo-env.sh",
               "DB_INTERF") ;

             // Extract value from line
```

```
                Pattern p = Pattern.compile(".*\" *(.*) *\".*");
                Matcher m = p.matcher(interf_act);
                m.find();
                String text_interf = m.group(1);

               // MET A JOUR IPV6 dans le fichier de
                  configuration si adresse IPv6 non PRESENTE
               if (DisplayName.equals(text_interf)){

                  String
                      ipv6_act=searchLine("/etc/profile.d/topo-env.sh",
                      "DB_IPV6") ;

                  // Extract value from line
                  p = Pattern.compile(".*\" *(.*) *\".*");
                  m = p.matcher(ipv6_act);
                  m.find();
                  String text_ipv6 = m.group(1);

                  if (text_ipv6.equals("")) {

                     // MODIFY DB_IPV6
                     out.printf(" * Modification Variable
                        environnement - Key DB_IPv6 : " + ip +
                        "\n");
                     executeCommand("DB_IPV6", ip);

                     // Duplicate file
                     cmdnet = " cp /etc/profile.d/topo-env.sh
                        /etc/profile.d/topo-env-dup.sh ";
                     runSystemCommand(cmdnet, "");


                  }
               }

             }

             out.printf("Index :" + idx + " - Name :" +
                NetArray.get(idx).GetName() + " - Time : " +
                NetArray.get(idx).GetInetTime() + " - Mac : " +
                NetArray.get(idx).GetMac() + "\n");

            //Tester si la valeur est toujours a true et non
               false, si oui
             process = true;

          } else {

             // NOUVELLE INTERFACE TROUVEE

            NetIntcount = NetIntcount + 1;

             ni.setName(DisplayName);
             ni.setCheck(rndNumber);
```

```
//Recuperation de l'adresse IPV6
// out.printf(" * IPv6 before:
   "+NetArray.get(i).GetadrIP()+"\n");
ni.setadrIP(ni.GetadrIP());
// out.printf(" * IPv6 after:
   "+NetArray.get(i).GetadrIP()+"\n");


String newipv6 = "";
String newmac="";

// newipv6= findIPv6(netint, "2001")

NetArray.add(ni);
out.printf("NEW Interfaces added : Name :" +
    DisplayName + "\n");


//
    ************************************************************
// SWITCHING ENTRE l ancienne interface et la
    nouvelle
// Basculement du routage de l'ancien vers le nouveau
// Recuperer l ancienne interface pour supprimer la
    regle


// Retourne l'interface plus ancienne dans la liste
    des interfaces disponibles
int i = findOldInterface(NetArray);

//out.printf("OLD INTERFACES index : " + i + "\n");

  if (i!= -1){

       // Recherche l'interface la plus ancienne
       String nameOldInterface =
           NetArray.get(i).GetName();
       String ipv6OldInterface =
           NetArray.get(i).GetadrIP();

      NetArray.get(i).setInetInitTime(LocalDateTime.now().toLocalTim

       DateTimeFormatter dtf =
           DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
       LocalDateTime now = LocalDateTime.now();
       String timechange = dtf.format(now);

       newipv6= findIPv6(netint, "2001");
       newmac =
           macAddress(netint.getHardwareAddress()) ;

       change =true ;
```

```
// MODIFY DB_IPV6
 out.printf(" * Modification Variable
    environmement - Key DB_MAC : "+newmac
    +"\n");
 executeCommand("DB_MAC",newmac);

 // MODIFY DB_IPV6
 out.printf(" * Modification Variable
    environmement - Key DB_IPv6 : "+newipv6
    +"\n");
 executeCommand("DB_IPV6",newipv6);

 //MODIFY DB_INTERFACE
 out.printf(" * Modification Variable
    environmement - Key DB_INTERF:
    "+DisplayName +"\n\n");
 executeCommand("DB_INTERF", DisplayName);

 out.printf(" * Modification Variable
    environmement - Key DB_IPv6_PREC:
    "+ipv6OldInterface +"\n");
 executeCommand("DB_IPV6_PREC",
    ipv6OldInterface);

 //MODIFY DB_INTERFACE
 out.printf(" * Modification Variable
    environmement - Key DB_INTERF_PREC:
    "+nameOldInterface +"\n");
 executeCommand("DB_INTERF_PREC",
    nameOldInterface);

 //MODIFY DB_INTERFACE
 out.printf(" * Modification Variable
    environmement - Key DB_TIME: "+timechange
    +"\n");
 executeCommand("DB_TIME", timechange);

 /*
 out.printf(" * DEL Default route ::/0 " +
    nameOldInterface + "\n");
 //cmdnet = "ip route del local ::/0 dev " +
    nameOldInterface ;
 //cmdnet = "ip -6 route del ::/0 " ;
 //runSystemCommand(cmdnet, "");

cmdnet = "ip -6 route del default" ;
runSystemCommand(cmdnet, "");
 */

 // out.printf(" * ADD Default route ::/0 to "+
    DisplayName + "\n");
//cmdnet = "ip route add local ::/0 dev ";
// runSystemCommand(cmdnet, DisplayName);


 out.printf(" * DISABLE OLD Interface : " +
```

```
                nameOldInterface + "\n\n");
            cmdnet = "ifconfig ";
            runSystemCommand(cmdnet, nameOldInterface + "
                down ");

            // Duplicate file
            out.printf(" * Replace File \n");
            cmdnet = " cp /etc/profile.d/topo-env.sh
                /etc/profile.d/topo-env-dup.sh";
            execCMD(cmdnet);


          /* CHANGE ROUTE FOR THE NEW INTERFACE */
          if (DisplayName.equals("b-eth0")){
             // cmdnet="ip -6 route replace default via
                2001:7ab:1::1 ";
             // runSystemCommand(cmdnet, "");

             cmdnet="ip -6 route replace ::/0 via
                2001:7ab:1:1:0:0:0:1 ";
             runSystemCommand(cmdnet, "");
          }

          if (DisplayName.equals("b-eth1")){
             // cmdnet="ip -6 route add ::/0 via
                2001:7ab:2::1 ";
             // runSystemCommand(cmdnet, "");

             cmdnet="ip -6 route replace ::/0 via
                2001:7ab:2:2:0:0:0:1 ";
             runSystemCommand(cmdnet, "");
          }

          if (DisplayName.equals("b-eth2")){
             // cmdnet="ip -6 route add ::/0 via
                2001:7ab:3::1 ";
             // runSystemCommand(cmdnet, "");

             cmdnet="ip -6 route replace ::/0 via
                2001:7ab:3:3:0:0:0:1 ";
             runSystemCommand(cmdnet, "");
          }

          out.printf(" * ADD Default route : "+ cmdnet +
             "\n");



        }else {


         // cmdnet = "ip -6 route del ::/0 " ;
         // runSystemCommand(cmdnet, "");


          //cmdnet = "ip -6 route del default" ;
```

```
                //runSystemCommand(cmdnet, "");

                //out.printf(" * ADD Default route ::/0 to "+
                    DisplayName + "\n \n");
                //cmdnet = "ip route add local ::/0 dev ";
                //runSystemCommand(cmdnet, DisplayName);

                //cmdnet = "ip route add ::/0 via ";

                /* CHANGE ROUTE FOR THE NEW INTERFACE */

                if (DisplayName.equals("b-eth0")){
                   // cmdnet="ip -6 route add ::/0 via
                       2001:7ab:1::1 ";
                   // runSystemCommand(cmdnet, "");

                   cmdnet="ip -6 route replace ::/0 via
                       2001:7ab:1:1:0:0:0:1 ";
                   runSystemCommand(cmdnet, "");

                }

                if (DisplayName.equals("b-eth1")){
                  // cmdnet="ip -6 route add ::/0 via
                      2001:7ab:2::1 ";
                  // runSystemCommand(cmdnet, "");

                   cmdnet="ip -6 route replace ::/0 via
                       2001:7ab:2:2:0:0:0:1 ";
                   runSystemCommand(cmdnet, "");
                }

                if (DisplayName.equals("b-eth2")){
                 // cmdnet="ip -6 route add ::/0 via
                     2001:7ab:3::1 ";
                 // runSystemCommand(cmdnet, "");

                   cmdnet="ip -6 route replace ::/0 via
                       2001:7ab:3:3:0:0:0:1 ";
                   runSystemCommand(cmdnet, "");
                }
                out.printf(" * ADD Default route : "+ cmdnet +
                    "\n");

                //out.printf(" * ADD Default route : "+ cmdnet
                    + "\n");
                // runSystemCommand(cmdnet, "");


            }


             process = true;

          }  // END ELSE
       } //END IF DisplayName equals LO
```

```
} // END FOR


// TESTER INTERFACES PRESENT DANS LE TABLEAU MAIS PLUS DANS
   LES INTERFACES : Interfaces down ou retires

/*************************************************************
 *           PART Variable Environnement      *
 *************************************************************/

// sed -i 's/export DB_IPV6=.*/export
   DB_IPV6="2001:8ab:820:1"/g' /etc/profile.d/topo-env.sh &&
   source /etc/profile.d/topo-env.sh && echo $DB_IPV6
// cmdnet= "cat /etc/profile.d/topo-env.sh | grep DB_IPV6 |
   sed 's/.*\"\\([^\"]*\\)\".*/\\1/'";

out.printf("\nRead Specific Environment Variable \n");


String mac=searchLine("/etc/profile.d/topo-env.sh", "DB_MAC")
   ;

// Extract value from line
Pattern p = Pattern.compile(".*\" *(.*) *\".*");
Matcher m = p.matcher(mac);
m.find();
String text = m.group(1);

out.printf(" Line MAC : " + text +"\n");


String interf=searchLine("/etc/profile.d/topo-env.sh",
   "DB_INTERF") ;

// Extract value from line
p = Pattern.compile(".*\" *(.*) *\".*");
m = p.matcher(interf);
m.find();
text = m.group(1);

out.printf(" Line INTERFACE : " + text +"\n");


String ipv6_act=searchLine("/etc/profile.d/topo-env.sh",
   "DB_IPV6") ;

// Extract value from line
p = Pattern.compile(".*\" *(.*) *\".*");
m = p.matcher(ipv6_act);
m.find();
text = m.group(1);

out.printf(" Line IPV6 : " + text +"\n\n");
```

```
String interf_prec=searchLine("/etc/profile.d/topo-env.sh",
    "DB_INTERF_PREC") ;

// Extract value from line
p = Pattern.compile(".*\" *(.*) *\".*");
m = p.matcher(interf_prec);
m.find();
text = m.group(1);

out.printf(" Line INTERFACE_PREC : " + text +"\n");


String ip=searchLine("/etc/profile.d/topo-env.sh",
    "DB_IPV6_PREC") ;
// Extract value from line
p = Pattern.compile(".*\" *(.*) *\".*");
m = p.matcher(ip);
m.find();
text = m.group(1);

out.printf(" Line IPV6_PREC : " + text +"\n");

String interf_time=searchLine("/etc/profile.d/topo-env.sh",
    "DB_TIME") ;
// Extract value from line
p = Pattern.compile(".*\" *(.*) *\".*");
m = p.matcher(interf_time );
m.find();
text = m.group(1);

out.printf(" Line INTERFACE TIME : " + text +"\n\n");


// GET Value of system environnement variable
//out.printf(" IPV6 : " + System.getenv("DB_IPV6")+"\n");


/*********************************************************
 *           PART CleanUP INFORMATION          *
 *********************************************************/
// Display number of interfaces founded
out.printf("CleanUp Operation \n");
NetArray= cleanUpInterface(NetArray, rndNumber);

/*********************************************************
 *           PART DISPLAY INFORMATION          *
 *********************************************************/
// Display number of interfaces founded
out.printf("\n");
out.printf("ArrayList size : %d\n", NetArray.size());

// Get the Enumeration object
Enumeration<NetInfo> e = Collections.enumeration(NetArray);

// Enumerate through the ArrayList elements
System.out.println("Content of ArrayList NetInfo "+ "");
```

```
        while(e.hasMoreElements()) {
           NetInfo ni = e.nextElement();

           out.printf(" - Name :" + ni.GetName() + " - Init Time : "+
              ni.GetInetInitTime() + " - Time : "+ ni.GetInetTime()
              + " - Mac : "+ ni.GetMac() + "\n");
           // out.printf(e.nextElement().GetInetTime() + "\n");
           //System.out.println(e.nextElement());

        }


        /*******************************************************
         *           PART SWITCHING INTERFACE      *
         *******************************************************/

        // Selon un nombre aleatoire < 20% et une duree superieure a
           5 second alors permettre un changement
        if (NetArray.size()>0){

           LocalTime mnt = LocalDateTime.now().toLocalTime();
           LocalTime init = NetArray.get(0).GetInetInitTime();

           // Ajoute a l heure d initiation + le delai et teste si
              superieure a l heure de maintenant pour autoriser le
              changement
           //init= init.plusSeconds(delaiSwitching);

           Duration duree ;
           long seconde = ChronoUnit.SECONDS.between(init, mnt);

           out.printf("\nTime init Interface : %s - Now : %s -
              Duration : %s \n",NetArray.get(0).GetInetInitTime(),
              mnt, seconde);

           //Generate random number for evaluate quality of the
              signal and then switching
           int signalrndNumber = 0 ;
           Random signalRand = new Random();
           signalrndNumber= signalRand.nextInt(100) + 1;
;

           out.printf("Current Signal : %s %%\n \n",signalrndNumber);



          // Changement d'Interface pour le transfert de paquets
          if (seconde > delaiSwitching && signalrndNumber<=20) {
              out.printf(" * Check Delay : %s and Signal : %s
                 %%\n",mnt, signalrndNumber);

              // Transmet le nom de l'interface actuel a la fonction
                 switchNode
             // et determine de maniere aleatoire la nouelle
                interface
```

```
                switchNode(NetArray.get(0).GetName());

            } //Fin IF


        }// FIN du IF NetArray


        /*******************************************************
         *           PART TESTING NETWORK          *
         *******************************************************/
        String ipv6 ="2001:4860:4860::8888";

       /*******************************************************
       *           PART DISABLE/ENABLE INTERFACES *
       *******************************************************/

       // Interrupt prcoess during 4 sec
       if (change==true){
          change=false ;
          sleep(delai+delaiChange);

       }else {
          sleep(delai);
       }


        cmdnet="clear";
        runSystemCommand(cmdnet," ");
        show=true;


    } // END OF WHILE

    // END OF MAIN
}


// Source :
    https://stackoverflow.com/questions/318239/how-do-i-set-environment-variables-f
public static void setEnv(String key, String value) {
    try {
       Map<String, String> env = System.getenv();
       Class<?> cl = env.getClass();
       Field field = cl.getDeclaredField("m");
       field.setAccessible(true);
       Map<String, String> writableEnv = (Map<String, String>)
           field.get(env);
       writableEnv.put(key, value);
    } catch (Exception e) {
       throw new IllegalStateException("Failed to set environment
           variable", e);
    }
}
```

```
public static String switchNode(String oldInterface) throws
    InterruptedException {

   boolean halte =true;
   Random rand = new Random();

   String strOldInterface = oldInterface.substring(0,
      oldInterface.length()-1);

   Integer oldval =
      Integer.parseInt(oldInterface.substring(oldInterface.length()-1,oldInterface

   Integer newval =-1;

   String cmdnet ="" ;
   String newInterface ="";

   // Nombre Interface possibles
   Integer minInterface = 0 ;
   Integer maxInterface = 2;

   while (halte){
      newval = rand.nextInt((maxInterface - minInterface) + 1) +
         minInterface;

      if (oldval != newval){
        halte=false;
      }

   }

   newInterface= strOldInterface + newval.toString();

   out.printf(" * Enabling New Interface - %s \n", newInterface);
   cmdnet = "ifconfig ";
   runSystemCommand(cmdnet, newInterface+ " up ");

   String newipv6 ="";

   //out.printf(" * Setting New Interface - %s \n", newInterface);
   /*  Affecte une adresse IPv6 a une interface */
   if (newInterface.equals("b-eth0")){
      newipv6= "2001:7ab:1:1:0:0:0:a/64";
   }

   if (newInterface.equals("b-eth1")){
       newipv6="2001:7ab:2:2:0:0:0:b/64";
   }

   if (newInterface.equals("b-eth2")){
      newipv6= "2001:7ab:3:3:0:0:0:c/64";
   }

   out.printf(" * Setting New IPv6 - %s \n", newipv6);
   cmdnet = "ifconfig " +newInterface +" inet6 add "+newipv6;
  // runSystemCommand(cmdnet, "");
```

```
    // cmdnet = "ip -6 route del ::/0 " ;
    // runSystemCommand(cmdnet, "");

     //cmdnet = "ip -6 route del default" ;
     //runSystemCommand(cmdnet, "");

    /* if (newInterface.equals("b-eth0")){
        cmdnet="ip -6 route add ::/0 via 2001:7ab:1::1 dev b-eth0";
     }

     if (newInterface.equals("b-eth1")){
        cmdnet="ip -6 route add ::/0 via 2001:7ab:2::1 dev b-eth1";
     }

     if (newInterface.equals("b-eth2")){
        cmdnet="ip -6 route add ::/0 via 2001:7ab:3::1 dev b-eth2";
     }
       */

     if (newInterface.equals("b-eth0")){
        //cmdnet="ip -6 route add default via 2001:7ab:1::1 dev
           b-eth0";
        cmdnet="ip -6 route add default via 2001:7ab:1:1:0:0:0:1";
     }

     if (newInterface.equals("b-eth1")){
        //cmdnet="ip -6 route add default via 2001:7ab:2::1 dev
           b-eth1";
        cmdnet="ip -6 route add default via 2001:7ab:2:2:0:0:0:1";
     }

     if (newInterface.equals("b-eth2")){
        //cmdnet="ip -6 route add default via 2001:7ab:3::1 dev
           b-eth2";
        cmdnet="ip -6 route add default via 2001:7ab:3:3:0:0:0:1";
     }

     out.printf(" * ADD Default route : "+ cmdnet + "\n");
     runSystemCommand(cmdnet, "");

     out.printf(" * Switching interface from %s to %s \n",
        oldInterface, newInterface);

     sleep(2000);
     return newInterface;

  }

  public static String searchLine(String path,String regexpr ) {
     File file = new File(path);
     String word = regexpr;
     String line ="";
     Scanner scanner = null;
```

```
        try {
            scanner = new Scanner(file);

        } catch (FileNotFoundException e) {
            //handle this
        }

        //now read the file line by line
        while (scanner.hasNextLine()) {
            line = scanner.nextLine();

            if (line.contains(word)) {
            //   System.out.println(line);
                scanner.close();
                return line ;
            }
        }
        scanner.close();
        return "";

    }

    public static void execCMD(String cmdline) throws IOException {


        String[] cmd = {"/bin/bash","-ic","echo password| sudo -S
            "+cmdline};


        Process pb = Runtime.getRuntime().exec(cmd);

        String line;
        BufferedReader input = new BufferedReader(new
            InputStreamReader(pb.getInputStream()));
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        input.close();
    }


    public static void executeCommand (String key , String value){
        ProcessBuilder processBuilder = new ProcessBuilder();

        // -- Linux --

        // Run a shell command
        processBuilder.command("bash", "-c", " sed -i 's/export "+ key
            +"=.*/export "+ key +"=\""+ value + "\"/g'
            /etc/profile.d/topo-env.sh");

         // Run a shell script
        //processBuilder.command("path/to/hello.sh");

        // -- Windows --
```

```
    // Run a command
    //processBuilder.command("cmd.exe", "/c", "dir
       C:\\Users\\mkyong");

    // Run a bat file
    //processBuilder.command("C:\\Users\\mkyong\\hello.bat");

    try {

       Process process = processBuilder.start();

       StringBuilder output = new StringBuilder();

       BufferedReader reader = new BufferedReader(
             new InputStreamReader(process.getInputStream()));

       String line;
       while ((line = reader.readLine()) != null) {
          output.append(line + "\n");
       }

       int exitVal = process.waitFor();
       if (exitVal == 0) {
         // System.out.println("Success!");
         // System.out.println(output);
         // System.exit(0);
       } else {
          //abnormal...
          System.out.println("Failed !");
          System.out.println(output);
       }

    } catch (IOException e) {
       e.printStackTrace();
    } catch (InterruptedException e) {
       e.printStackTrace();
    }
}


/**
 * Execute a bash command. We can handle complex bash commands
    including
 * multiple executions (; | && ||), quotes, expansions ($), escapes
    (\), e.g.:
 *    "cd /abc/def; mv ghi 'older ghi '$(whoami)"
 * @param command
 * @return true if bash got started, but your command may have
    failed.
 */

static ArrayList<NetInfo> initArrayNet (ArrayList<NetInfo>
   NetArray) throws SocketException {
   /*****************************************
    *          PART INITZIALIZATION *
    *****************************************/
```

```
    // Retrieve information about interfaces
    Enumeration<NetworkInterface> nets =
       NetworkInterface.getNetworkInterfaces();

    //Enumeration<NetworkInterface> netscount =
       NetworkInterface.getNetworkInterfaces();
    //out.printf("InetAddress found : %s\n\n",
       Collections.list(netscount).size());
    int NetIntcount =0;

    //Initialization & Interate for each interface retrievd
    for (NetworkInterface netint : Collections.list(nets)) {
       NetInfo ni = new NetInfo();

       ni= displayInterfaceInformation(netint, ni, false );
       String DisplayName =
          netint.getDisplayName().toString().replace(" ","");

       if (!DisplayName.equals("lo")) { // IF Interface is not
          loopback
          NetIntcount = NetIntcount + 1;

          ni.setName(DisplayName);
          NetArray.add(ni);
       }
    }
    return NetArray ;
}



static int findOldInterface ( ArrayList<NetInfo> nilist){

    int indexItem =-1 ;
    LocalTime dateInterface = null;

    // Enumerate through the ArrayList elements
    System.out.println("\nFinding old Interface of ArrayList NetInfo
       "+nilist.size() );

    for (int i = 0; i < nilist.size(); i++) {
       if (i==0){
          indexItem=0;
          dateInterface= nilist.get(i).GetInetTime();
       }

       if (nilist.get(i).GetInetTime().isBefore(dateInterface) ){

          //out.printf("Index :" +indexItem+ " - Name :" +
             nilist.get(i).GetName() + " - Time : "+
             nilist.get(i).GetInetTime() + " - Mac : "+
             nilist.get(i).GetMac() + " - Check : "+
             nilist.get(i).GetCheck() + "\n");

          dateInterface= nilist.get(i).GetInetTime();
          indexItem= i;
```

```
        }
    }
    return indexItem;
}

static ArrayList<NetInfo> cleanUpInterface ( ArrayList<NetInfo>
    nilist, int rnd){
    int indexItem =-1 ;

    for (int i = 0; i < nilist.size(); i++) {

        indexItem= i;

        out.printf("Index :" +indexItem+ " - Name :" +
            nilist.get(i).GetName() + " - Time : "+
            nilist.get(i).GetInetTime() + " - IP : "+
            nilist.get(i).GetadrIP() + " - Check : "+
            nilist.get(i).GetCheck() + "\n");
    }

    // Get the Enumeration object
   // Enumeration<NetInfo> e = Collections.enumeration(nilist);

    // Enumerate through the ArrayList elements
    System.out.println("\nCleanUP of ArrayList NetInfo "+ "");

    ArrayList<NetInfo> nicopy = new ArrayList<NetInfo>();

    for (NetInfo nl : nilist){
        if (nl.GetCheck()!= rnd){
          // nilist.remove(nl);
            out.printf(" Remove : Name :" + nl.GetName() + " - Time :
                "+ nl.GetInetTime() + " - Mac : "+ nl.GetMac() + "\n");

        }
        else {
            NetInfo ni = nl.CastNetInfo(nl);
            nicopy.add(ni);
        }

    }


  /* for (Iterator i = nilist.listIterator(); i.hasNext();) {
        NetInfo ni = i.next();
        if (ni.GetCheck()!= rnd){
            out.printf(" Remove : Name :" + ni.GetName() + " - Time :
                "+ ni.GetInetTime() + " - Mac : "+ ni.GetMac() + "\n");
            i.remove();
        }
    }*/
    //return nilist;
    return nicopy;
}

static int searchInterface ( ArrayList<NetInfo> nilist, String
```

```
        name){
        int indexItem =-1 ;

        for (int i = 0; i < nilist.size(); i++) {

            if (nilist.get(i).GetName().equals(name)){
                indexItem= i;

                //out.printf("Index :" +indexItem+ " - Name :" +
                    nilist.get(i).GetName() + " - Time : "+
                    nilist.get(i).GetInetTime() + " - Mac : "+
                    nilist.get(i).GetMac() + "\n");
                return indexItem ;
            }

        }

        return indexItem;
    }

    static NetInfo displayInterfaceInformation(NetworkInterface netint,
        NetInfo ni, boolean info) throws SocketException {

        int NetAdrcout=0;

        //out.printf("Name: %s\n", netint.getName());
        Enumeration<InetAddress> inetAddresses =
            netint.getInetAddresses();

        if (netint.isUp()) {
            String DisplayName =
                netint.getDisplayName().toString().replace(" ","");

            if (!DisplayName.equals("lo")) { // IF Interface is not
                loopback

                if (info==true){ out.printf("\nDisplay name: %s
                    (UP:%s)\n", DisplayName, netint.isUp()); }

                for (InetAddress inetAddress :
                    Collections.list(inetAddresses)) {
                    NetAdrcout= NetAdrcout+1;

                    String ip = inetAddress.toString().replace("/", "");

                    if (isIPv4(ip))
                    {
                      if (info==true){ out.printf("InetAddress IPv4:
                          %s\n", ip); }
                    }
                    else {
                       try {
                           String[] split = ip.split("%");

                           String firstSubString = split[0];
                           String secondSubString = split[1];
```

```
                //out.printf("InetAddress: %s\n", ip);
                if (info==true) {
                   out.printf("InetAddress IPv6: %s\n",
                      firstSubString);
                }

                String st ="";
                st = firstSubString.substring(0,4);

                // IF adr IPv6 begin to 2001, save IPv6
                /* if (st.equals("2001")) {
                   ni.setadrIP(firstSubString);
                   out.printf("Save IPv6 :" + ni.GetadrIP() +
                      "\n");
                }
                */

                //out.printf("InetAddress Interface: %s\n",
                   secondSubString);


             } catch (Exception ex) {
                if (info==true){ out.printf("InetAddress Ex:
                   %s\n", ip);}

             }

          }
       }

       //out.printf("Up? %s\n", netint.isUp());
       ni.setUp(netint.isUp());

       //out.printf("Loopback? %s\n", netint.isLoopback());
       ni.setLoopback(netint.isLoopback());

       //out.printf("PointToPoint? %s\n",
          netint.isPointToPoint());
       //out.printf("Supports multicast? %s\n",
          netint.supportsMulticast());
       //out.printf("Virtual? %s\n", netint.isVirtual());
       if (info==true){ out.printf("Hardware address: %s\n",
          macAddress(netint.getHardwareAddress())); }
       ni.setMac(macAddress(netint.getHardwareAddress()));

       //      Arrays.toString(netint.getHardwareAddress()));
       //out.printf("MTU: %s\n", netint.getMTU());
       if (info==true) {
          out.printf("Inet Address found : %d\n", NetAdrcout);
          out.printf("\n");
       }
    }
}
```

```
        // Return all interfaces UP
        return ni;
    }

    public static String findIPv6(NetworkInterface netint, String
        pattern )throws SocketException {


        int NetAdrcout=0;
        String ipAdr = "";

        //out.printf("Name: %s\n", netint.getName());
        Enumeration<InetAddress> inetAddresses =
            netint.getInetAddresses();

        if (netint.isUp()) {
            String DisplayName =
                netint.getDisplayName().toString().replace(" ", "");

            if (!DisplayName.equals("lo")) { // IF Interface is not
                loopback

                for (InetAddress inetAddress :
                    Collections.list(inetAddresses)) {
                    NetAdrcout = NetAdrcout + 1;

                    String ip = inetAddress.toString().replace("/", "");

                    if (isIPv4(ip)) {
                        String ok = "";
                    } else {
                        try {
                            String[] split = ip.split("%");

                            String firstSubString = split[0];
                            String secondSubString = split[1];

                            //out.printf("InetAddress: %s\n", ip);
                           // out.printf("InetAddress IPv6: %s\n",
                              firstSubString);

                            String st = "";
                            st = firstSubString.substring(0, 4);

                            // IF adr IPv6 begin to 2001, save IPv6
                            if (st.equals(pattern)) {
                                ipAdr = firstSubString;
                             //  out.printf("Save IPv6 :" + ipAdr + "\n");
                            }

                            //out.printf("InetAddress Interface: %s\n",
                                secondSubString);
                        } catch (Exception ex) {
                            out.printf("InetAddress Exception: %s\n", ip);
                        }
                    } // FIIN ELSE
```

```
            } // FIN FOR
         }// FIN IF lo
      } // FIN IF isUP
      return ipAdr;
   }


   // Source : https://gist.github.com/madan712/4509039
   public static boolean isReachable(String address) {
      try {
         InetAddress inetAddress = InetAddress.getByName(address);
         boolean isReachable = inetAddress.isReachable(5000);
         System.out.printf("Is the address [%s] reachable? -%s\n",
            address, isReachable ? "Yes" : "No");
         return isReachable;

      } catch (Exception e) {
         e.printStackTrace();
      }
      return false;

   }

   //
      https://bytenota.com/java-how-to-check-if-an-ip-address-is-ipv4-or-ipv6/
   public static boolean isIPv4(String ipAddress) {
      boolean isIPv4 = false;

      if (ipAddress != null) {
         try {
            InetAddress inetAddress = InetAddress.getByName(ipAddress);
            isIPv4 = (inetAddress instanceof Inet4Address) &&
               inetAddress.getHostAddress().equals(ipAddress);
         } catch (UnknownHostException ex) {
         }
      }

      return isIPv4;
   }

   public static boolean isIPv6(String ipAddress) {
      boolean isIPv6 = false;

      if (ipAddress != null) {
         try {
            InetAddress inetAddress = InetAddress.getByName(ipAddress);
            isIPv6 = (inetAddress instanceof Inet6Address);
         } catch (UnknownHostException ex) {
         }
      }

      return isIPv6;
   }

   public static String macAddress(byte[] mac) {
      try {
```

```
            //System.out.print("Current MAC address : ");

            StringBuilder sb = new StringBuilder();
            if (mac != null) {
               for (int i = 0; i < mac.length; i++) {
                  sb.append(String.format("%02X%s", mac[i], (i <
                     mac.length - 1) ? "-" : ""));
               }
               //System.out.println(sb.toString());
            }
            return sb.toString();

         } catch (Exception e) {

            //e.printStackTrace();
            return "";

         }
      }


      // Source : https://gist.github.com/madan712/4509039
      public static void runSystemCommand(String command, String ipv6) {


         //System.out.printf("Running : "+ command +" " +ipv6 +"\n");
         try {
            Process p = Runtime.getRuntime().exec(command+ " " + ipv6);
            BufferedReader inputStream = new BufferedReader(
                  new InputStreamReader(p.getInputStream()));

            String s = "";
            // reading output stream of the command
            while ((s = inputStream.readLine()) != null) {
               System.out.println(s);
            }

         } catch (Exception e) {
            e.printStackTrace();
         }
      }

}
```

```
import java.time.LocalTime;

public class NetInfo{

   private String inetName;
   private String inetMac;

   private int typeIP ;
   private String adrIP;
   private LocalTime inetTime ;
```

```
private LocalTime inetInitTime ;

private boolean isLoopback;
private boolean isUp;
private boolean isReacheable;
private int check;


// Constructor
NetInfo () {
   this.adrIP="";
   this.inetName ="";
   this.inetMac ="";
   this.inetTime = LocalTime.now();
   this.inetInitTime = LocalTime.now();
   this.typeIP =6;

   this.isLoopback = false;
   this.isUp = false;
   this.isReacheable=false;
   this.check=0;


}


//public NetInfo CastNetInfo(String inetName, String inetMac, int
   typeIP, String adrIP, LocalTime inetTime, boolean isLoopback,
   boolean isup, boolean isReacheable, int check){
   public NetInfo CastNetInfo(NetInfo nic){

   NetInfo ni = new NetInfo();

  ni.setName(nic.GetName());
  ni.setMac(nic.GetMac());
  ni.setTypeIP(nic.GetTypeIP());
  ni.setadrIP(nic.GetadrIP());
  ni.setInetTime(nic.GetInetTime());
  ni.setInetInitTime(nic.GetInetInitTime());
  ni.setLoopback(nic.GetLoopback());
  ni.setUp(nic.GetUp() );
  ni.setReacheable(nic.GetReacheable());
  ni.setCheck(nic.GetCheck());

  return ni;
}

public void setName(String name){
   this.inetName=name;
}

public String GetName(){
   return this.inetName;
}

public void setInetTime(LocalTime localtime){
```

```
        this.inetTime=localtime;
    }

    public LocalTime GetInetTime(){
        return this.inetTime;
    }


    public void setInetInitTime(LocalTime localtime){
        this.inetInitTime=localtime;
    }

    public LocalTime GetInetInitTime(){
        return this.inetInitTime;
    }


    public void setadrIP(String adrIP){
        this.adrIP=adrIP;
    }

    public String GetadrIP(){
        return this.adrIP;
    }


    public void setMac(String mac){
        this.inetMac=mac;
    }

    public String GetMac(){
        return this.inetMac;
    }

    public void setLoopback(boolean loopback){
        this.isLoopback=loopback;
    }

    public boolean GetLoopback(){
        return this.isLoopback;
    }

    public void setTypeIP(int typeIP){
        this.typeIP =typeIP;
    }

    public int GetTypeIP(){
        return this.typeIP;
    }

    public void setUp(boolean up){
        this.isUp=up;
    }

    public boolean GetUp(){
        return this.isUp;
```

```
    }


    public void setReacheable(boolean reacheable){
        this.isReacheable=reacheable;
    }

    public boolean GetReacheable(){
        return this.isReacheable;
    }

    public void setCheck(int check){
        this.check=check;
    }

    public int GetCheck(){
        return this.check;
    }



   /* @Override
    public String toString() {
        return String.format("Name: %s | Phone Number: (%d)-%d",
                name, areaCode, phoneNumber);
    }*/
}
```